Neue Serie: Einführung in Unix

mc

6

Die Mikrocomputer-Zeitschrift

7 DM · 60 öS · 7 sfr. · Juni 1987

Universelles Programmiergerät:

PALs, EPROMs, PROMs

Betriebssysteme:
OS-9 für Atari ST

PC-Technik:
Digitalisieren

Grundlagen:
Linienalgorithmus
Integerarithmetik
Datenbanken

LC-Anzeige
für Basic-EMUF

Liebe Leser, fleißige Biber, Eins und Null!

Da habe ich mich in die Nesseln gesetzt, mit meinem Editorial in Ausgabe 5! Leider war da nicht nur einiges mißverständlich, sondern sogar falsch. Gott sei Dank ist der Biberwettbewerb noch nicht zu Ende, sonst hätte ich ihn verdorben. Das klassische Biberproblem besteht natürlich darin, daß man eine Turingmaschine auf das leere Band ansetzt, sie arbeiten läßt und abwartet, wieviele Einsen sie produziert. Dabei kann die Reihe der Einsen doch durchaus mit Nullen durchsetzt sein! Ich habe da eine Literaturstelle schuldhaft falsch interpretiert und muß die Biberzüchtergemeinde um Vergebung bitten. Natürlich gibt es auch einen fleißigsten Biber mit 5 Zuständen, weil es davon ja nur endlich viele gibt. Soweit meine Missetaten.

Aus den vielen Anrufen und Briefen konnten wir großes Interesse registrieren. Ich hoffe, daß jetzt stabile Verhältnisse im Wettbewerb eingetreten sind. Der Hauptpreis, ein Amiga 2000, ist für den schnellsten klassischen Biber mit 5 Zuständen ausgeschriebenen, wie in Ausgabe 4 auch geschehen. Mein Fehler im Editorial zu Ausgabe 5 hat aber eine Variante angesprochen, die gar nicht so unsinnig ist.

Während das klassische Problem sehr energisch von George Uhing (New York) angegangen wurde, indem er einen Biber mit 1915 Einsen fand (der auch Nullen enthält), hat noch niemand Biber mit ununterbrochenen Strecken von Einsen untersucht. Damit niemand umsonst gedacht und gerechnet hat, gibt es einen wertvollen Trostpreis für die Einsendung, die die längste Einsenstrecke produziert. Dabei darf rundherum alles mögliche stehen. Insgesamt gilt sowohl für den klassischen Biber, als auch für die Variante mit der Einsenstrecke , daß bei gleicher Länge die Rechenzeit entscheidet und wenn das keinen Unterschied hergibt, das Los. Ich bin gespannt, wieviele Einsendungen in der Endrunde noch eingehen. Der Wettbewerb endet am 30. Juni 1987. Einsendungen, die einen späteren Poststempel besitzen, können nicht mehr berücksichtigt werden.

Soweit zu den Bibern. In dieser Ausgabe gibt es einen besonderen Leckerbissen für alle Hardware-Profis. „mc-Allprog" haben wir das Gerät getauft, das so universell einstellbar ist, daß es nahezu alle programmierbaren Standard-ICs programmieren kann, ob es PALs, PROMs oder EPROMs sind. Wir sind stolz auf die dreiteilige Serie, die den Aufbau und die Programmierprinzipien schildert. Zu diesem Bauvorschlag gibt es von den Autoren einen preiswerten Bausatz und ein Softwarepaket, das sogar einen PAL-Assembler-Editor bietet. In dieser Ausgabe läuft außerdem eine Einführung in das Betriebssystem Unix an, geschrieben von einer Praktikerin, die ihr Wissen schon in vielen Kursen an EDV-Profis weitergegeben hat.

Ihr

Ulrich Rohde

mc-inhalt

mc-editorial

Liebe Leser
Fleißige Biber, Eins und Null! 3

mc-briefe 6

mc-info 8

Know-how-Computer 49

Echtzeit-Betriebssystem für den PC 103

Bilder mit dem Mac 119
Grafikgestaltung für Profis

Spruch des Monats 112

mc-bücher 32

mc-hard

mc-AllProg 40
So umfangreich das Angebot an festprogrammierbaren Bausteinen ist, so unterschiedlich sind auch die Programmier-Vorschriften. Wenn man nicht ständig die gleiche Art von Bausteinen einsetzt, muß man immer wieder neu teure Geräte kaufen. Wir bringen ein universelles Programmiergerät für PALs, EPROMs und PROMs, das auch die Einstellungen vornimmt

Z80-Karte für den mc-68000-Computer 50
Im zweiten Teil der Beschreibung unserer Z80-Karte wird die Inbetriebnahme, Fehlersuche und die prinzipielle Arbeitsweise der CP/M-80-Emulator-Software behandelt

Basic-EMUF mit LCD und Tasten 60
Ein gutes Beispiel dafür, daß mc-Projekte leben und gepflegt werden, ist der Basic-EMUF, den wir in mc 12/86 vorstellten. Diesmal zeigen wir, wie an den leistungsfähigen Computer eine separate Tastatur und LC-Anzeige angeschlossen werden

NDR- und mc-CP/M-Computer: IBM-kompatibel 66
Auf das umfangreiche Angebot an Software- und Hardware-Erweiterungen, die für IBM- und kompatible Computer angeboten werden, können jetzt der NDR- und der mc-CP/M-Computer zugreifen. In einem System arbeiten NDR- und Intel-CPU parallel

Digitalisieren mit „Kompatiblen“ 75
Ein Leckerbissen für alle PC-Anwender ist dieser Beitrag, der eine universelle Ein-/Ausgabeeinheit bringt. Und damit sie ihre Leistungsfähigkeit unter Beweis stellen kann, wird der mc-Digitalisierer angeschlossen

TERM1 mit 19 200 Baud 83
Ein Tip für alle mc-CP/M-Computer-Freunde

14 MByte RAM auf einer Steckkarte – 80386 für 8088-PCs 123

mc-grundlagen

Einführung in Unix 86
Obwohl über das Betriebssystem Unix schon viel geschrieben wurde, werden Sie in unserer Serie viele neue interessante Informationen finden. Ein Beitrag nicht nur für Profis

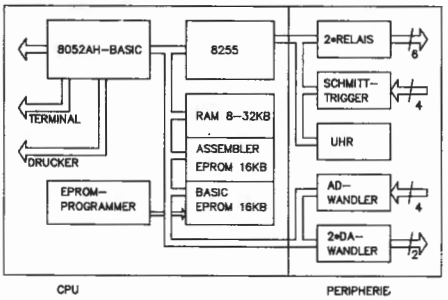
Titelfoto: Klaus Hager

mc-AllProg

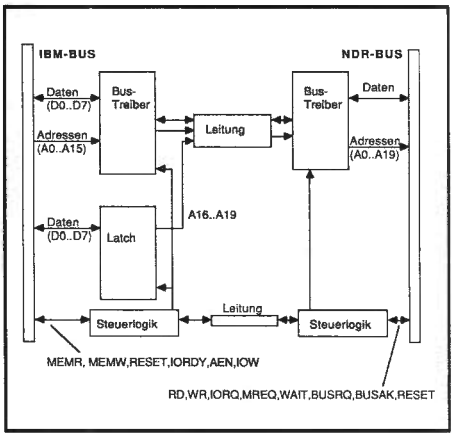
Übertreibungen und reißerische Aufmachungen gehören nicht zu unseren Arbeitsmitteln. Auf unser neuestes Projekt, mit dem PALs, EPROMs und PROMs programmiert werden können und zu dem ein PAL-Editor gehört, sind wir zu Recht stolz. Womit das begründet wird, lesen Sie ab **Seite 40**

Basic-EMUF mit LCD und Tasten

Der Einsatz eines professionellen Einplatinen-Computers macht erst dann so richtig Spaß, wenn man während des Betriebs direkt Zustände abfragen und Eingaben machen kann. Wie das an unserem erfolgreichen Basic-EMUF funktioniert, lesen Sie ab **Seite 60**



BLOCKSCHALTBILD BASIC–EMUF



NDR- und mc-CP/M-Computer: IBM-kompatibel

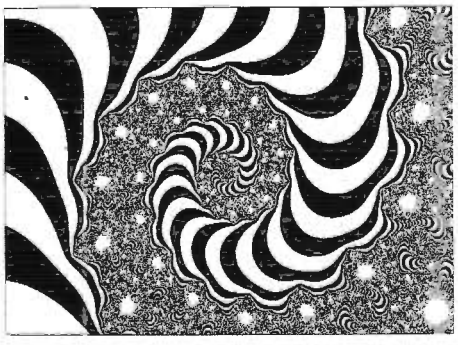
Der NDR-Computer und der IBM-PC können verbunden werden. Zwei Platinen, eine auf jeder Seite, konvertieren die Bus-Signale. So kann der IBM-Kompatible in den NDR-Computer hineinschauen – oder der NDR-Computer mit dem IBM-kompatiblen Nachrichten austauschen. Das Motto lautet: Zusammenarbeit. Das geht sogar soweit, daß der IBM-kompatible Computer das NDR-System als Parallelrechner benutzen kann.

Seite 66

Mandelbrot-Menge mit 8087

Vor der Erforschung der interessanten Gebiete in der Mandelbrot-Menge steht immer wieder die Geduldsprobe, bis der Rechner mit den Berechnungen fertig ist. Wie ein Co-Prozessor für schnellere Ergebnisse sorgt, beschreiben wir ab

Seite 120



GOTO und Standard-Basic 92
GOTO oder nicht GOTO – das ist hier die Frage. Das Für und Wider untersucht ein Beitrag, der jeden Programmierer zum Mitmachen auffordert und interessante Denkanstöße gibt

Datenbanksysteme 96
Das richtige Datenbanksystem kann nur derjenige finden, der weiß, worauf er bei seiner Suche achten muß. Ein Beitrag, der Fehler vermeiden hilft

Wahrheitstafeln für Logik-PROMs 100
Mit einem Programm in Turbo-Pascal werden Wahrheitstafeln für PROMs erzeugt

Schneller Drawline-Algorithmus 104
Noch immer soll es Programmiersprachen geben, die keinen Befehl zum Linienzeichnen kennen. Wir erklären leichtverständlich, worauf bei der Implementation einer entsprechenden Routine zu achten ist

mc-soft

Siemens PC-D mit hoher Auflösung 94
Eine Assembler-Routine zur Ansteuerung der hochauflösenden Grafik

Routine zum Übernehmen von Dateinamen in Strings 99
Nützliche GW-Basic-Routine für zahlreiche Anwendungen

Sin und cosin mit Integerarithmetik 108
Für den Z80 stellen wir Assembler-Routinen vor, die Werte von Winkelfunktionen schnell und genügend genau berechnen

Menüauswahl 114
Auf Komfort braucht man auch bei selbstgeschriebenen Programmen nicht verzichten. Wie Funktionen aus einer Auswahlmenge übersichtlich gewählt werden, zeigt eine Turbo-Pascal-Prozedur

GEM für alle PCs 116
Wenn GEM – die grafische Benutzeroberfläche – auf Ihrem PC nicht so will wie Sie, hilft Ihnen vielleicht dieser Beitrag weiter

Mandelbrot-Menge schnell berechnen 120
Erst mit einem mathematischen Co-Prozessor kommt die rechte Freude bei der Berechnung von Mandelbrot-Mengen auf einem PC auf

mc-test

Producer 122
Datenbankprogramm programmieren lassen?
Warum nicht, wenn es so einfach geht wie mit Producer

OS-9 für den Atari ST 125
Das Multiuser-/Multitasking-System wurde für den Atari ST angepaßt und läuft zufriedenstellend. Wenn es Anwendungs-Software unter OS-9 für den Atari ST gäbe, könnte es sich auch auf breiter Basis durchsetzen

mc-markt 130

mc-vorschau 166

Impressum 164

mc ist...

Die Ausgabe 5/87 von mc ist – kurz gesagt – spitze! Nach den letzten Ausgaben glaubte ich schon, diese tolle Fachzeitschrift wird zu einem Reklameblättchen der Industrie („nach Unterlagen von xyz").

Da ich zur Zeit einen Z80-Slaveprozessor für IBM PCs entwickle, war für mich der Artikel über die „Z80-Karte im mc-68000-Computer" besonders interessant. Da ich den Z80 nicht als CP/M-Rechner, sondern als Drucker-Spooler- und V-24-Subsystem einsetzen will, von CP/M aber verhältnismäßig wenig Ahnung habe, würde ich eine genaue Beschreibung dieser Implementation sehr begrüßen.
Schließlich eine Anmerkung zum Artikel „Cross-Assemler und EPROM" (mc 5/87, S. 74): HEX-Files lassen sich auch mit Debug in EPR-Files umwandeln, da dieses ein HEX-File bereits beim Einlesen umwandelt und das CX-Register (gelesene Bytes) richtig setzt.

A. Landenberger
8000 München 40

68008-Karte läuft auf dem Apple IIe

Sie läuft doch – auch auf dem Apple-IIe! Gemeint ist die in mc 9/85 vorgestellte 68008-Karte. Als Besitzer eines IIe-kompatiblen Rechners wollte ich den Einstieg in die „68000-Welt" riskieren und bestellte ungeachtet der Warnungen des Franzis-Software-Services die Karte. Nach dem Bestücken (die Karte wies einige Leiterbahnfehler auf) und der Inbetriebnahme legte die Karte los und lief einwandfrei.
Nun frage ich mich, warum der Verlag potentiellen Nachbauern den Zugang erschwert hat, indem jahrelang – anscheinend ohne angemessen zu prüfen – verbreitet wurde, sie liefe nur auf dem II+-Rechner.
Überhaupt gibt mir die Politik des Verlags Rätsel auf. Meiner Meinung nach ist der Apple immer noch ein guter Entwicklungsplatz für Software, z. B. für den 68008-EMUF. Bisher aber gab es diesbezüglich nur einen Beitrag in mc – und zwar in Form eines Leserbriefs.
Betreibt etwa der Verlag hier Protektionismus für eigene Produkte, wie den 68008-EMUF?
Ich bin mir sicher, daß ich mit meinem Wunsch, doch in Zukunft mehr Beiträge in punkto Apple-Software zu bringen, nicht allein dastehe.

Andreas Löwe
5270 Gummersbach

Anm. der Red.: Unsere Warnung, daß die 68008-Karte im Apple-IIe Probleme macht, heißt ja nicht, daß die Karte dort mit Sicherheit nicht läuft: Wie immer bei Zeitproblemen genügen oft allein die Bauteilestreuungen, um einmal problemlos zu laufen, einmal nur ab und zu abzustürzen und im dritten Rechner den Dienst ganz zu verweigern. Seriöse Firmen weisen ihre Kunden auf solche Probleme hin – wie eben der Franzis-Software-Service...

„Kosmetik" für Grafik-Dump-Routine

Die von Gerhard Mehrke in der Ausgabe 4/87 auf Seite 97 vorgestellte Grafik-Dump-Routine für Apple II-Rechner mit Epson-Interface hat zwei Schönheitsfehler, die zu unliebsamen Überraschungen führen können:

1. Die ROM-Routinen des APL-Interface benutzen einen Teil des 6502-Input-Buffers ($200-$2FF) als Zwischenspeicher. Unter CP/M wird dieser Bereich ($F200-$F2FF) zwar normalerweise nicht benutzt, allerdings legen hier einige Zusatzkarten ihre eigene Treibersoftware ab. Dies gilt beispielsweise für die Saturn-RAM-Karten oder den Ehring Controller. Nach Aufruf der Graphikroutine sind schließlich einige Speicherstellen überschrieben und die dort angelegten Programme produzieren nur noch Unsinn. Daher muß dieser Bereich zuerst in einen Zwischenspeicher gerettet und anschließend zurückkopiert werden.

2. Das Graphik-Dump-Register muß innerhalb der Dump-Prozedur mit den entsprechenden Werten belegt werden, da sowohl die Initialisierungroutine des ROMs als auch die Dump-Routine selbst diesen Wert überschreibt. Am besten übergibt man daher die gesamte Information an die Druckroutine.
Das korrigierte Listing ist im *Bild* zu sehen.
Last but not least ein Hinweis: Im Handbuch wurde nicht erwähnt, daß ein Grafik-Dump problemlos durch Eingabe von Ctrl-C vorzeitig beendet werden kann.

Peter Engels
5307 Wachtberg-Niederbachem

```
PROCEDURE HARDCOPY (Enlarge, Inverse, Ddens: Boolean);

(* fuer graphikfaehiges Interface in Slot 1; z.B. Epson- oder
 Apple-Parallelkarte *)

VAR MODE     : BYTE ASOLUTE  $F779;
    DDBYTE   : BYTE ASOLUTE  $F479;
    A_REG    : BYTE ABSOLUTE $F045;
    BUFFER   : ARRAY (.$00..$7F.) OF BYTE;

BEGIN
   MOVE(MEM(.$F280.),BUFFER,$80);
   A_REG := 0;
   CALL6502 ($C100);
   MODE   := 2 + BYTE (ENLARGE) * 64 + BYTE (INVERSE) * 32;
   DDBYTE := 75 + BYTE(DDENS);
   A_REG  := $91;
   CALL6502($C102);
   MOVE(BUFFER,MEM(.$F280.),$80)
END;
```

Das korrigierte Listing der Grafik-Dump-Routine

(In-)Kompatibilität als Fortschritt?

Als früherer Kolumnen-Schreiber möchte ich zum Thema Kompatibilität als Zeichen des Fortschritts, das mc in der Ausgabe 4/87 behandelte, Stellung nehmen:
Der im mc-editorial sinngemäß unter dem Motto „Kompatibilität – insgeheim doch Fortschritt?" festgestellte Widerspruch zu einer früheren Kolumne wird schon auf S. 39 („Ist IBM zu sich selbst noch kompatibel?") ad absurdum geführt. Wo bitte, ist da der angebliche Fortschritt von IBM? Wenn Kompatibilität kein Fortschritt ist, wie ich einmal behauptete, dann ist allein die Tatsache, daß IBM mit sich selbst inkompatibel ist, doch noch kein Fortschritt! Denn eine IBM-Harddisk ist in keiner Weise schneller und besser als eine XYZ-Harddisk.

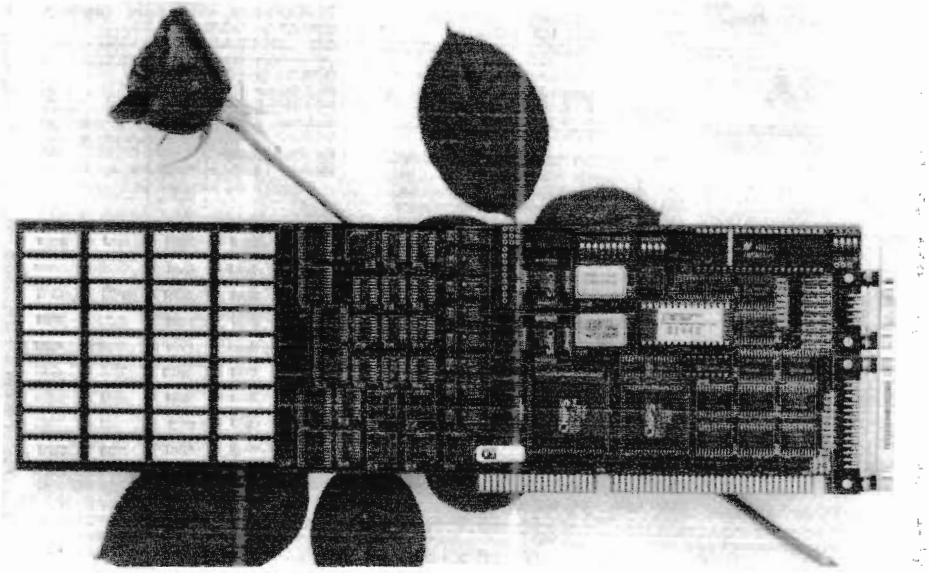
Herwig Feichtinger
8000 München 40

EGA-Grafik mit Zutaten

Auf PC-Zusatzkarten ist die in Bodenheim bei Frankfurt ansässige Firma Computer Elektronik Infosys (CEI) spezialisiert. Die Produkte werden in Deutschland entwickelt und gefertigt. Zu den jüngst vorgestellten Neuheiten gehören zwei Karten, die unter anderem die EGA-Grafik beherrschen.

Die EGA-2 wird mit 256 KByte Bildspeicher geliefert. Ihre Betriebs-Software befindet sich schon „an Bord", zusätzliche Programme müssen deshalb nicht geladen werden. Im Farbmodus sind Auflösungen von 320 × 200 bis 640 × 350 Bildpunkten möglich. Grafik und Text lassen sich dabei mischen. Ebenso kann man in der Monochrom-Betriebsart mit 640 × 350 Bildpunkten Grafik und Text gemischt darstellen. In der höchsten Auflösung, 720 x 350 Bildpunkte, ist nur Monochrom-Textdarstellung möglich. Ein Zeichensatz mit maximal 512 Zeichen läßt sich per Software laden. Von den insgesamt 64 möglichen Farben können gleichzeitig 16 dargestellt werden. Gewissermaßen als Zugabe erhält der Käufer der 1083 DM teueren Karte eine parallele Druckerschnittstelle.

Wer mit dem Speicher seines AT oder IBM-XT286 nicht mehr auskommt, ist mit der Multifunktionskarte Multi-4-AT gut bedient. Sie nimmt bis zu 4 MByte RAM auf, und zwar gemäß der Lotus-/Intel-Spezifikation. Das heißt, Programme wie Lotus 1-2-3, Symphony und Framework benutzen diesen Speicher. Zusätzlich stehen auf dieser Karte alle Funktionen der EGA-Grafik sowie eine serielle und eine parallele Schnittstelle zur Verfügung. Warte-Zyklen fügt diese Platine erst bei Taktfrequenzen über 8 MHz ein. Mit 2 MByte RAM bestückt kostet die Multi-4-AT-Karte rund 3050 DM. Für weitere 2 MByte muß man gut 650 DM mehr hinlegen.



Eine EGA-kompatible Grafik und bis zu 4 MByte RAM bietet die Multi-4-AT-Karte von CEI

SHAMROCK Software Vertrieb liefert Franzis-Software

Die Aktivitäten des bisherigen Franzis-Software-Service werden seit dem 1. Mai 1987 von der neu gegründeten SHAMROCK Software Vertrieb GmbH weitergeführt. Dieses Unternehmen wurde gemeinsam vom Franzis-Verlag und der Shamrock Software GmbH gegründet. Wie vielen unserer Leser bekannt ist, gehört die Shamrock Software GmbH dem ehemaligen mc-Chefredakteur Herwig Feichtinger und Alfred Schön, der ebenfalls viele Jahre als Redakteur an der mc mitwirkte. Die beiden halten über ihre bisherige Firma die Mehrheitsbeteiligung am neuen Unternehmen.

Für unsere Leser wird sich nicht viel ändern: Der neue Software-Vertrieb wird auch weiterhin zu fairen Preisen und in enger Zusammenarbeit mit den Redaktionen Programme aus Franzis-Zeitschriften und -Büchern in Form von Sammeldisketten anbieten. Ein weiterer Schwerpunkt sind Programme für technisch-wissenschaftliche Anwendungen sowie System-Software, Programmierwerkzeuge und Utilities für Software-Entwickler. Ziel des Shamrock Software-Vertriebs ist es, die Produkte innerhalb von 24 Stunden nach Bestelleingang auszuliefern. Außerdem soll das Händlernetz erheblich ausgeweitet werden, damit Interessenten Software auch vor Ort testen können. Eine Mailbox soll die Kommunikation mit den Kunden oder verschiedener Programmbenutzer untereinander intensivieren (Tel.-Nr. 0 89/ 59 64 65, Parameter 300,8,N,1, 18 bis 7 Uhr).

Die bisherige Telefonnummer (0 89/51 17-3 31) hat sich nicht geändert, die neue Anschrift lautet Karlstraße 35, 8000 München 2.

Novell-Netz für IBMs System/2

Kaum sind IBMs neue Computer der Serie System/2 angekündigt, da melden sich schon die ersten Hersteller, die dafür Produkte anbieten wollen. Mit am schnellsten hat die Firma Novell reagiert. Sie will die neuen Geräte voll in ihre Netzwerk-Software „NetWare" einbinden. In Verbindung mit diesem LAN-Betriebssystem lassen sich die neuen Modelle als LAN-Workstations oder File-Server einsetzen. Alle Modelle können künftig auch in bestehenden lokalen Netzen als Workstations fungieren.

EGA, CGA und Hercules auf einer Karte

Impecs neue Grafikkarte Vision7 entspricht in allen wesentlichen Eigenschaften dem EGA-Standard, die Auflösung ist mit 640 × 480 Punkten allerdings höher. In dieser Darstellung, bei der gleichzeitig 16 Farben aus einer Palette von 64 angezeigt werden können, ist das Produkt kompatibel zum „IBM-Professional-Graphics-Adapter". Eine erweiterte Textdarstellung mit 60 Zeilen zu je 80 Zeichen erhöht insbesondere die Übersicht bei Tabellenkalkulationsprogrammen. Neben dem EGA-Modus emuliert die Karte auch die CGA- und die Hercules-Karte.

Den Entwurf eigener Zeichensätze unterstützt die mitgelieferte Software. Sieben Zeichensätze (darunter einer für APL) sind auf der zum Lieferumfang gehörenden Diskette schon abgespeichert. Außerdem bekommt der Kunde ein Installationsprogramm, Diagnose-Software und das Zeichenprogramm Dr. Halo. Die Karte kostet 1220 DM.

Optische Platten auch löschbar

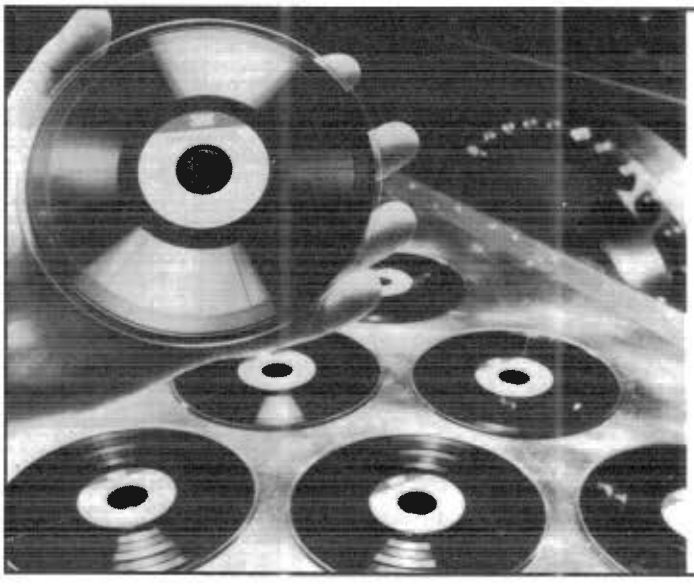
Optischen Speichermedien wird eine glänzende Zukunft vorausgesagt. Allein in den USA, so schätzt Frost & Sullivan in einer Studie, wird man im Jahr 1991 mit Laufwerken und Platten rund 1,3 Mrd. $ umsetzen. Bis dahin sollen Laufwerke jährlich um 80 % und optische Medien um 100 % zulegen. Daß sich die optimistischen Prognosen für die Gegenwart nicht erfüllt haben, liegt nach Ansicht der Marktforscher an den Problemen mit der Löschbarkeit optischer Platten. Die könnten aber bald behoben sein. Immerhin haben jetzt mehrere Firmen die ersten serienreifen Produkte angekündigt.

Zu diesen Firmen gehören 3M und Verbatim. Beide wenden das thermo-magnetische optische Verfahren an, das allgemein mit TMO (Thermo Magneto Optical Disk Technology) abgekürzt wird. Es läßt Spurdichten von 16 000 tpi (Spuren pro Zoll) zu. Das entspricht Speicherkapazitäten von 200...400 MByte pro Seite einer 5¼-Zoll-Diskette (3M bietet dieses Format an). Verbatim bringt auf seinen 3,5-Zoll-Disketten 50 MByte unter. Wie die herkömmlichen magnetischen Disketten stellen auch die neuen Datenträger ein auswechselbares Medium dar.

Bei den neuartigen Platten liegt unter einer lichtdurchlässigen Schicht eine reflektierende Schicht. Das lichtdurchlässige Material gerät bei Erwärmung in einen instabilen Zustand, den sich das TMO-Verfahren zunutze macht. Ein schwaches Magnetfeld ist nämlich dann in der Lage, die magnetische Polarität des erwärmten Bereichs zu ändern. Trifft polarisiertes Licht auf einen derartigen „Magnetic Spot", dann wird die Polarisationsebene des Lichts dort gedreht, wo auch die „Magnetrichtung" umgepolt ist. Der Schreibvorgang geht also so vor sich: Ein Laserstrahl relativ hoher Intensität erwärmt das Material, und ein Magnetfeld legt die Polarisationsrichtung fest.

Gelesen wird mit einem Laserstrahl geringer Intensität, der von der reflektierenden Schicht zu einem Polarisationsfilter gelangt. Je nach Polarisationsrichtung läßt das Filter den Strahl durch oder blockt ihn ab: Das Bit wird von der hinter dem Filter angebrachten Empfangsdiode als 0 oder 1 erkannt. Auch der Löschvorgang ist recht einfach: Dabei erwärmt wieder der „kräftige" Laserstrahl die reflektierende Schicht, und ein Magnetfeld richtet alle „Bits" in derselben Richtung aus.

Der größte Vorteil dieser Aufzeichnungsmethode liegt in der Datensicherheit. Bei rein magnetischen Medien gibt es eine Reihe von Fehlerquellen, die hier entfallen. So gibt es keinen Abrieb durch Schreib-/Lese-Köpfe, weil die entsprechenden Vorrichtungen einige Millimeter von der Plattenoberfläche entfernt sind. Auch Staub- und Schmutzpartikel können dem Medium kaum etwas anhaben. Umgelegt auf die Kapazität des Datenträgers sind die Kosten der neuen Technik wesentlich geringer. Allerdings sind die Laufwerke selbst noch recht teuer.



Nach dem TMO-Verfahren wird diese optische Platte von 3M beschrieben und gelöscht

Turbo-Freunde, aufgepaßt!

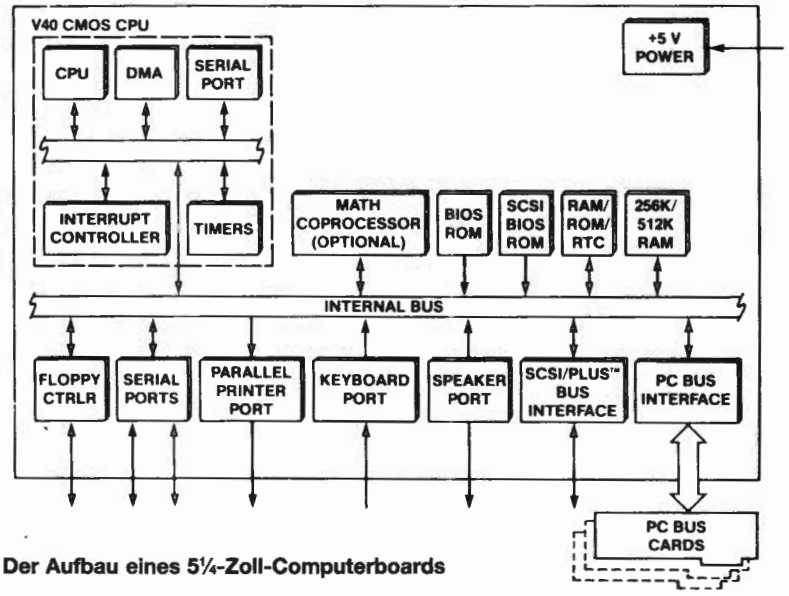
Eine Reihe von Neuigkeiten gibt es jetzt wieder für Turbo-Pascal-Programmierer. Eine davon heißt Turbo-Inliner und kommt von der Frankfurter Firma Totalinformation GmbH. Dabei handelt es sich um eine Art Quelltextgenerator, der aus einem Assembler-Programm Hochsprachenquelltext in Form von Inline-Anweisungen erzeugt. Dadurch bleibt dem Programmierer das mühsame und fehlerträchtige Abtippen von Hexadezimalcodes erspart. Eine mitgelieferte Batch-Datei sorgt für das automatische Zusammenspiel zwischen Assembler, Linker und Inliner. Das Programm kostet 98 DM und ist auch in einer Modula-2-Version erhältlich.
SPEED-LIB nennt sich eine Programm-Bibliothek für Turbo-Pascal, deren Besonderheit darin besteht, daß sie den erforderlichen Speicherplatz vom Heap abzweigt. Im Code-Segment, in dem das Programm liegt, beansprucht sie lediglich etwa 5 KByte für die Turbo-Pascal-Schnittstelle. Die Bibliothek besteht aus einer Sammlung von über 100 Prozeduren und Funktionen, die die Entwicklung „schlüsselfertiger" Programme erleichtern. Das Programmpaket besteht unter anderem aus einem Maskeneditor sowie aus Routinen zum bildschirmorientierten Editieren von Strings oder Feldern. Um zu demonstrieren, wie man die Bibliothek anwendet, befindet sich der Quelltext des Maskeneditors auf der Programmdiskette. Das Programmpaket wird vom Verlag Heinz Heise, Hannover, zu einem Preis von 148 DM angeboten.
Erweiterungen zu Turbo-Pascal kommen auch von der deutschen Borland-Vertretung Heimsoeth Software. Sie heißen schlicht „BCD", „8087" und „Maus". Das erste Produkt steigert die Rechengenauigkeit und ist vor allem bei kaufmännischen Anwendungen von Nutzen. Das zweite sorgt für höhere Geschwindigkeit, wenn der Coprozessor 8087 vorhanden ist. Wer Programme schreiben will, die mit einer Microsoft-kompatiblen Maus bedient werden, für den ist das letztgenannte Produkt das richtige. Jedes der drei Programmpakete kostet rund 112 DM.

Laserdrucker preiswerter

Die Okidata GmbH hat ihre Preise für den auf der CeBIT vorgestellten Laserdrucker Laserline 6 Plus um 10 % gesenkt. Der Laserline 6 Plus mit 512 KByte wird jetzt für 4998 DM angeboten.

Single Board Computer

Das amerikanische Unternehmen Ampro, in Deutschland vertreten durch die IST Elektronik GmbH aus Unterhaching, stellt eine Reihe von Single Board Computerplatinen und ausbaufähige Systeme für Industrieanwender vor. Basierend auf den Prozessoren NEC V40, 80186 oder Z80 A stehen mehrere komplette Systeme zur Wahl. Für jede der CPUs ist ein eigenes Projektboard-Entwicklungssystem erhältlich, welches die Entwicklung von speziell anwendungsbezogenen Systemen extrem erleichtert. Als Betriebssystem wird PC-DOS oder Concurrent-DOS eingesetzt. Der onboard-Speicher ist bis auf 768 KByte ausbaubar. Die Schnittstellen des Boards, das in 5¼-Zoll Standard-Abmessungen ausgeführt ist, sind aus dem Bild ersichtlich. Dort sind auch eine Reihe von Zusatzplatinen eingezeichnet.



Der Aufbau eines 5¼-Zoll-Computerboards

Mailbox liefert kostenlos Software

Kostenlos Software können sich Computer-Freaks jetzt per Telefon ins Haus holen. Die Hamburger Firma Dr. Neuhaus hat im letzten Jahr eine Mailbox installiert, die Platz für durchschnittlich 200 Programme (20 MByte) bietet. Nach anfänglichen Schwierigkeiten läuft das System nach Auskunft der Firma jetzt problemlos. Die Mailbox enthält Programme für CP/M- und MS-DOS-Rechner. Sie steht jedem offen, der das intelligente Modem Smarty der Firma Dr. Neuhaus besitzt. Warum andere Modems ohne zusätzliche Software nicht geeignet sind, liegt ganz einfach daran, daß die Mailbox ein Datensicherungsprotokoll voraussetzt, das Smarty bereits von sich aus beherrscht. Prinzipiell lassen sich aber auch Fremd-Modems verwenden, wenn dieses Protokoll per Software realisiert wird. Unterlagen darüber versendet die Firma auf Anforderung kostenlos. Die technischen Voraussetzungen sind ähnlich wie bei Btx (1200 und 75 Bd).
Und hier noch das Wichtigste: Die Mailbox ist unter folgenden Telefonnummern zu erreichen: 040/580341, 0251/719964, 06151/333151, 0911/607085.

Kostenlos kann man sich Programme jetzt aus einer Mailbox überspielen: Am einfachsten geht das mit dem intelligenten Modem Smarty

Computer-Treff

Vom Bayerischen Fernsehen wird im dritten Programm einmal im Monat ein Computermagazin für Laien und Freaks ausgestrahlt, der Computer-Treff. In der Serie von E. Huber, R. D. Klein und J. Plate wird über Themen aus den verschiedensten Bereichen der Computertechnik und -Anwendungen berichtet. Das Magazin bringt Beiträge über aktuelle Hard- und Software und erklärt, wie sie funktionieren. In den letzten Folgen wurde zum Beispiel über Musik mit dem Computer, Sprachausgabe und -erkennung und künstliche Intelligenz berichtet. Für diejenigen, die gerne basteln, gibt es Tips und Bauanleitungen. Im Januar wurde z. B. ein Steuersystem für eine Modellbahn gezeigt. Zu jeder Sendung gibt es ein kostenloses Begleitheft, in dem die wichtigsten Dinge zusammengefaßt und Literaturhinweise gegeben werden. Gesendet wird der Computer-Treff immer am frühen Nachmittag des letzten Sonntags im Monat. Die Juni-Sendung ist am 28. 6. 87 um 13 Uhr 43.

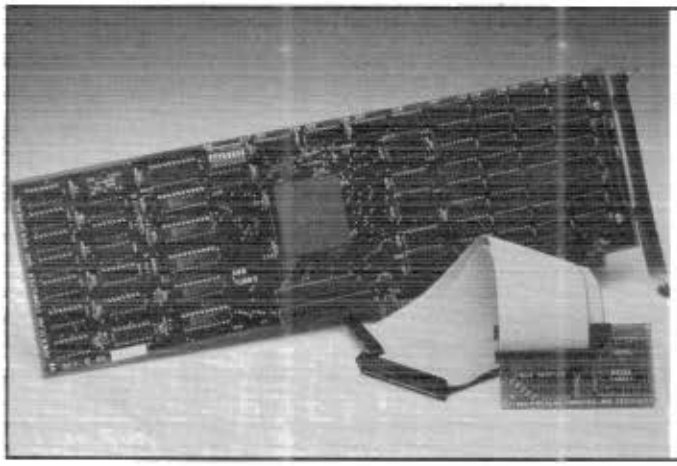
Sammelordner gratis

In den bekannten Sonderheftreihen des Franzis-Verlages, die sich aus interessanten Artikeln der FUNKSCHAU und der ELEKTRONIK zusammensetzen, finden sich thematisch sortierte Grundschaltungen der Elektronik sowie Schaltungsideen für Elektroniker. Bei Bestellungen von mindestens fünf Sonderheften der Reihe „Schaltungsideen für Elektroniker" oder „Grundschaltungen der Elektronik" bekommt man jetzt einen stabilen Sammelordner kostenlos dazu.

80386-Beschleuniger

Die Fast Card 386 ist eine einsteckbare Beschleunigungskarte zum Ausbau von IBM AT und Vollkompatiblen zu einem echten 80386-Rechner. Die von der Taunussteiner Firma Norbert Dixius Fast Machines gelieferte 386-Karte verbessert das Zeitverhalten von IBM ATs um bis zu 400 % bei völliger Kompatibilität zur existierenden AT-Software. Die Karte verdoppelt die Taktfrequenz eines mit 6 MHz getakteten AT auf 12 MHz und beschleunigt einen 8-MHz-AT auf 16 MHz. Die gewünschte Frequenz wird über eine mitgelieferte Treibersoftware eingestellt. Auf dem Board der 386 Fast Card befindet sich ein Cache-Speicher von 1 MByte, in dem alle Operationen des 80386 zwischengespeichert werden. Neben dem ROM-BIOS wird auch das gesamte Betriebssystem in diesen Speicher geladen. Dies führt dazu, daß ein AT mit 4 MIPS arbeitet. Die Fast Card wird in einen Steckplatz des ATs gesteckt, ein spezieller Adapter wird in die Fassung des nun überflüssigen 80286-Prozessors gesteckt. Der Preis der gesamten Systemerweiterung inklusive Adapter, Software, Einbauanleitung und Chipzange liegt bei 3500 DM.

Ähnlich aufgebaut, jedoch auf einer halben Karte untergebracht ist die Fast Card 286, die zum Ausbau eines PC XT oder Kompatiblen zum AT gedacht ist.



Mit dieser Steckkarte kann man aus den meisten ATs einen vollwertigen 80386-PC machen

AT-Portable von Osborne

Der mit tragbaren CP/M-Computern bekannt gewordene Hersteller Osborne hat jetzt zwei Portable der AT-Klasse vorgestellt, den OCC-08-D mit zwei Diskettenlaufwerken (720 KByte) und den OCC-08-H mit einem Floppylaufwerk und einer 20-MByte-Festplatte. Beide Modelle arbeiten mit einer 8-MHz-80286 CPU und sind mit 640 KByte RAM (ohne Wartezyklen) ausgestattet. Das 9 kg schwere Modell mit zwei Diskettenlaufwerken kostet 6486 DM, die Festplattenversion 7686 DM. Über eine Erweiterungsbox sind IBM PC/AT-Steckkarten adaptierbar. Neben DOS 3.2 und Basic ist den Osborne-Computern noch das integrierte Paket Micropro Easy beigelegt.

Einer der preiswertesten AT-Portable kommt von Osborne

Training bei Motorola

Im Rahmen des umfangreichen Kursprogramms, das der Motorola Geschäftsbereich Halbleiter anbietet, sind in einigen Kursen noch Plätze frei. Vom 01.06.87 bis zum 04.06.87 wird ein Kurs über das Peripheriekonzept des MC68000 abgehalten. Über die Software zum 68020 kann man sich vom 03.06.87...05.06.87 informieren. Wer sich für die Coprozessoren interessiert, sollte sich für den Kurs vom 22.06.87...24.06.87 einschreiben. Im Juni gibt es weitere Kurse zum 68000, 68020, RMS68K und Sysgen. Einzelheiten zu den Kursen erfährt man unter der Telefonnummer 089/9272142, wo man sich auch anmelden kann.

Z80-Nachfolger ist da

Der seit längerer Zeit als Z800 angekündigte 16-Bit-Nachfolger des Z80 ist jetzt in Musterstückzahlen erhältlich. Allerdings trägt der neue Baustein jetzt die Bezeichnung Z280. Er kann Software, die für den Z80 geschrieben wurde, ohne jede Modifikation ausführen. Die derzeitigen Muster laufen mit einer internen Taktfrequenz von 10 MHz. Das führt zu einer mindestens vierfachen Prozessorleistung gegenüber dem Z80A.

Für einen hohen Durchsatz sorgen auch eine dreistufige Pipeline und ein 256-Byte-Cache für Befehle und Daten. Eine integrierte Speicherverwaltungseinheit, die nach dem Paging-Verfahren arbeitet, adressiert bis zu 16 MByte physikalischen Speicher und vergibt Zugriffsrechte für 4 oder 8 KByte große Speicherseiten. Dies macht sich vor allem bei Multitasking-Systemen positiv bemerkbar.

Neben dem Taktoszillator enthält der Z280 eine asynchrone serielle Vollduplex-Schnittstelle, vier DMA-Kanäle, drei 16-Bit-Zähler/-Timer, einen DRAM-Controller und Wait-State-Generatoren. Der Bus läßt sich auch so betreiben, daß er zu dem des Z80 kompatibel ist.

EPROMs schnell, als Set oder parallel!

EPROM Produktions-Programmer:

- für EPROMs, EEPROMs und CMOS-PROMs bis 512 K
- "Intelligent" und "High Speed" Programmierung
- einfache und sichere Bedienung

Halbleiter-Hersteller Freigaben

MINATO 1867A

- 8-fach-Parallel-Programmer
- EPROM-Prüfung vor und nach der Programmierung
- Checksum- und Fehleranzeige
- kompakt, 2 kg leicht
- Service-Koffer lieferbar

DM 3.950,–
+ 14% MwSt. 553,–
DM 4.503,–*

STOLLCON ST-161

- 16-fach Programmer
- Parallel- und Set-Programmierung
- Zugriffszeitmessung
- "Silicon Signature" Funktion
- interaktive Bedienerführung
- Tastatur-Verriegelung
- Datenspeicher 256 KB
- RS-232C Schnittstelle

ab DM 9.950,–
+ 14% MwSt. 1.393,–
DM 11.343,–*

UV-Löschgeräte

- für Einzel-EPROMs und EPROM-Boards

ab DM 240,–
+ 14% MwSt. 33,60
DM 273,60*

Direktkontakt:
Fr. Schmieder 089/4208-142
*unverbindliche Preisempfehlung

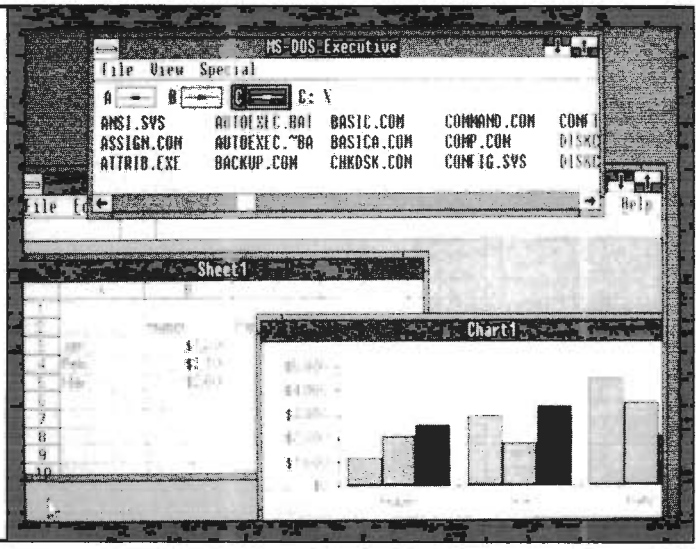
MACROTRON

Stahlgruberring 28 · 8000 München 82 Tel. (089) 4208-0
Tx. 529448 mato · Teltex 897280 = mato · Tfax 089-429563

mc-info

Microsoft kündigt Windows 2.0 an

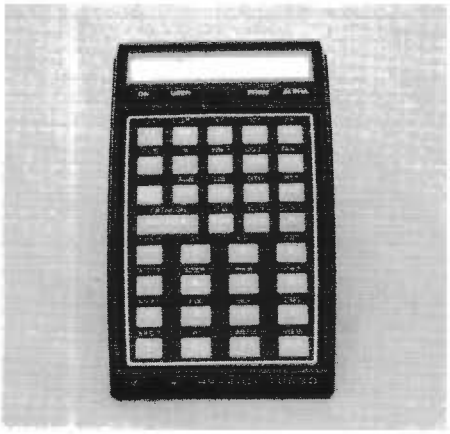
Windows 2.0 stellt eine Brücke von MS-DOS zu MS-OS/2 dar. Die grafische Bedieneroberfläche ist diesselbe wie im MS-OS/2-Windows-Presentation-Manager. Mit Windows 2.0 sind überlappende Fenster möglich. Bisher für Windows geschriebene Anwendungen werden auch auf der neuen Version lauffähig sein. Windows 2.0 wird im dritten Quartal im Handel erhältlich sein.



Fenstertechnik auf dem PC mit Windows 2.0

HP-41 getunt

Auf der Basis des Taschenrechners HP-41 von Hewlett-Packard entwickelte die Fa. W&W Software Products aus Bergisch Gladbach den Rechner HP-41CY Turbo. Dieser Rechner unterscheidet sich von dem bekannten HP-41CX durch einen Zusatzspeicher von 64 KByte sowie durch eine um den Faktor 2 gesteigerte Rechengeschwindigkeit und ein verändertes Design. Im Zusatzspeicher laufen alle Programme ohne Probleme ab. Besonders als hochwertiger Datensammler eignet sich der erweiterte Speicher, der auch bei Batteriewechsel keine Daten verliert. Zusätzliche 8 KByte Betriebssystem erlauben Programmierung in Maschinensprache. Alle HP-41-Rechner, die nach Juli 1986 gefertigt wurden, können nachgerüstet werden. Ein HP-41CY Turbo kostet etwa 1300 DM.



Der HP-41-CY Turbo bietet 64 KByte RAM und ist doppelt so schnell wie ein normaler HP-41

Bürokommunikationssysteme auf der Systems

Immer mehr deutsche Unternehmen rüsten ihre Arbeitsplätze mit Bürokommunikationssystemen aus. Nach einer Prognose der Marktforscher der Managementberatungsfirma Diebold werden bis 1991 etwa 90 % aller Großunternehmen und etwa 20 % der mittleren und kleinen Unternehmen Bürokommunikationssysteme einsetzen. Bis zum Anfang dieses Jahres lag die Zahl der installierten Systeme bei 4000; bis 1990 ist mit mehr als 25 000 Systemen zu rechnen. Darin sieht die Münchener Messe- und Ausstellungsgesellschaft die sachliche Rechtfertigung für einen speziellen Ausstellungsbereich „Systeme für den Arbeitsplatz Büro" auf der in diesem Jahr vom 19...23. Oktober stattfindenden Systems.

Mailbox-Programm besteht FTZ-Prüfung

Wer das Mailbox-Programm PCMAIL von Shamrock Software verwendet, kann jetzt sicher sein, daß damit die V.24-Schnittstelle korrekt bedient wird. Nach einer entsprechenden Prüfung durch das Fernmeldetechnische Zentralamt der Deutschen Bundespost (FTZ) hat das Programm eine Zulassungsnummer erhalten. Gekoppelt ist diese Zulassung allerdings an den IBM-PC-kompatiblen Rechner PC-7500 von Sharp.

Laut dieser Post-Zulassung ist der Betrieb an Datex-L- (300 Bit/s) und Datex-P-Hauptanschlüssen (300/1200 Bit/s), im Direktrufnetz (300/1200 Bit/s) sowie im öffentlichen Fernsprechnetz (300/1200 Bit/s) zulässig. Damit entspricht die Konfiguration den CCITT-Normen V.21, V.22, V.25, X.20bis und X.28 (P20) für asynchrone Kommunikation.

PCMAIL wird von vielen Firmen vorwiegend zur Kommunikation mit Außendienst-Mitarbeitern, Händlern und Kunden eingesetzt. Das Programm unterstützt sowohl Post-Modems als auch FTZ-zugelassene Modem-Boxen (z. B. Smarty). Die aktuelle Version 2.0 für 485 DM erlaubt neben dem Einrichten privater Briefkästen für mehrere hundert Anrufer auch Benutzergruppen. Das heißt, Dateien lassen sich bestimmten Benutzernamen individuell zuordnen.

Multitech zieht nach

Unter die Anbieter von 80386-Computern hat sich jetzt auch das taiwanesische Unternehmen Multitech eingereiht. Generalimporteur für den neuen MPC-1100 ist die Ahrensburger Firma CE-TEC. Das System läßt sich auf insgesamt sechs Taktfrequenzen von 4,77...16 MHz einstellen. Dabei benötigt es auch bei der höchsten Taktfrequenz keine Wartezyklen. Sämtliche MS-DOS-Programme laufen nach Angaben der Vertriebsfirma problemlos. Mit der Einführung neuer Betriebssysteme für derartige Computer wird auch die Frage des Speicherausbaus wieder aktuell. Der MPC-1100 kann insgesamt 16 MByte adressieren. Auf der Hauptplatine haben 2 MByte Platz, ein spezieller 32-Bit-Steckplatz steht für Erweiterungen zur Verfügung. Daneben gibt es zwei Steckplätze mit 32-Bit-Bus, drei mit 16-Bit-Bus und zwei mit 8-Bit-Bus. Die beiden letztgenannten Kategorien sind kompatibel zum PC- bzw. AT-Bus. Zwei RS-232C-Schnittstellen und ein Centronics-Interface für den Drucker gehören zur Grundausstattung. Ebenso eine interne Uhr und eine Grafikkarte, die CGA- und Hercules-kompatibel ist.

Mit einem 1,2-MByte-Laufwerk, 1 MByte Hauptspeicher und 40-MByte-Festplatte (Zugriffszeit 28 ms) kostet der MPC-1100 rund 13 000 DM. In diesem Preis ist auch ein 14-Zoll-Monitor enthalten. Wer 2 000 DM mehr ausgibt, erhält ein System mit EGA-Karte und geeignetem Monitor.

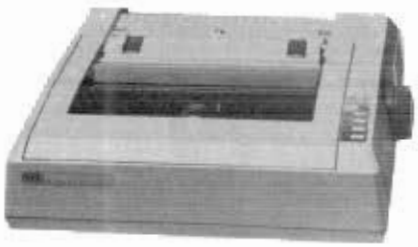
Ein weiterer Computer, der als CPU den Prozessor 80386 enthält: der MPC-1100 von Multitech

NEC

Der Handliche

NEC Pinwriter P6

- 24 Nadel-Präzisionsdruckknopf (2x12 versetzt)
- Geringe Geräuschentwicklung (53 dBA im Quiet Mode)
- Auflösung von 360x360 Punkten/Zoll
- Druck in diversen Standardschriften, bis doppelte Höhe und dreifache Breite
- Bis 220 Zeichen/Sek.
- Auch in Colorversion erhältlich



Coupon

mc 6/87

☐ Bitte senden Sie mir mehr Informationsmaterial
☐ Bitte senden Sie mir Ihre Firmenzeitschrift DISPLAY

Name _____

_____

Straße _____

PLZ/Ort _____

Tel.: _____

Computer-Peripherie-Geräte GmbH
6455 Erlensee · Beethovenstraße 26
Telefon 0 61 83/83-0

Modula 2 Compiler für Atari ST umtauschen

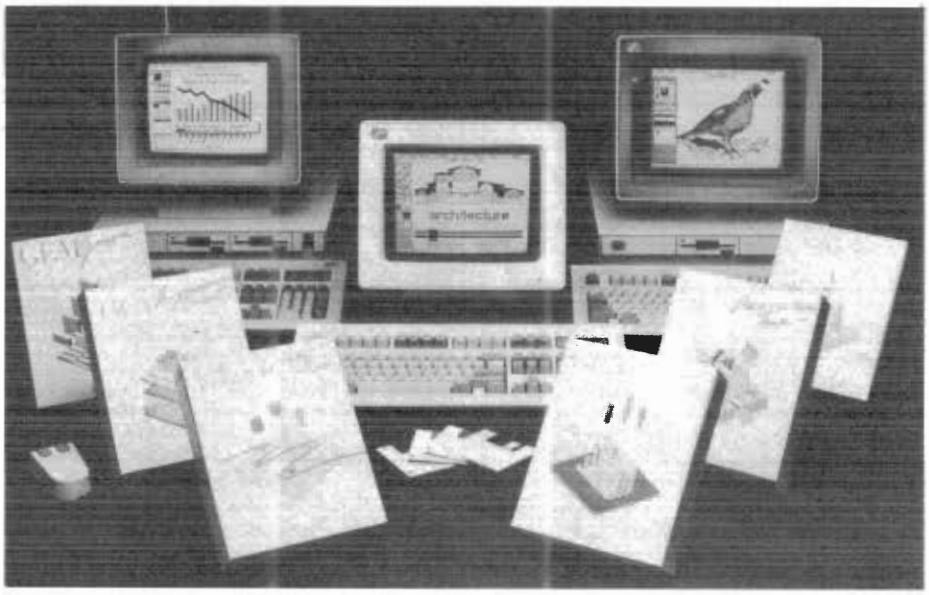
Modula 2 und das Toolkit für den Atari ST liegt jetzt in einer neuen Version (3.0) vor. Der volle Speicherbereich des Prozessors wird jetzt unterstützt. Aus urheberrechtlichen Gründen wird das Update für etwa 30 britische Pfund nur in England vorgenommen. Wegen des umständlichen Versands übernimmt diesen die Firma Knupe (0231/527531) für ihre Kunden in einer Sammelaktion zu einem Unkostenbeitrag von 10 DM. Das neue Modula ist ab sofort zu einem Preis von 298 DM zu beziehen.

Preissenkung für CD-ROM Laufwerke

Mit einer Preissenkung von fast 40% stimuliert die Hitachi Sales Europe GmbH in Hamburg die Installations-Entwicklungen im CD-ROM Markt. Das Hitachi CD-ROM Laufwerk CDR-1503S, das mit einem Audioausgang für multimediale CD-ROM Anwendungen ausgerüstet ist, kostet jetzt etwa 2850 DM und damit etwa 1650 DM weniger als vor der Preissenkung. Die erste multimediale CD-ROM Anwendung und die erste integrierte Anwendung einer CD-ROM zeigte Hitachi auf der Infobase in Frankfurt. Bei der multimedialen Anwendung handelte es sich um eine Enzyklopädie, bei der der Anwender neben der gewohnten Textinformation auf Grafik-, Bild- und Toninformationen zugreifen kann. Als integrierte CD-ROM Anwendung wurde „Bookshelf" gezeigt. Auf einer CD sind dabei verschiedene Nachschlagwerke für den Schriftgebrauch am Schreibtisch wie Grammatik, Satzbau und Stil gespeichert. Dazugehörige Software erlaubt dem Anwender, die abgerufenen Informationen direkt in die Textverarbeitung am Bildschirm zu übernehmen, ohne das Programm verlassen zu müßen. Beide CDs wurden von Microsoft entwickelt.

GEM-Applikationen für IBM PS/2

Digital Research teilte mit, daß ab sofort alle GEM-Applikationen für die neuen Computer der /2-Familie von IBM verfügbar sind. Programme wie GEM Draw Plus, GEM Wordchart und das GEM Programmers Toolkit werden auf 3½-Zoll-Disketten angeboten. Erste Tests sollen beeindruckende Leistungsmerkmale gezeigt haben. Besonders die gesteigerte Geschwindigkeit der Computer soll sich erheblich bemerkbar machen. Die GEM-Applikationen unterstützen sowohl den Standard- als auch den hochauflösenden Grafikmodus.



Von Anfang an liefert Digital Research GEM-Software für die neuen Computer der PS/2-Familie von IBM

Termine

INTERACT '87, die zweite Konferenz über „Human-Computer Interaction" findet vom 01.09.87...05.09.87 in Stuttgart statt. Informationen hierzu gibt es bei Interact '87, c/o IPA/IAO, Nobelstrasse 12, Postfach 800469, 7000 Stuttgart 80.
Bei der Messe Stuttgart kann das Programm für den Anwenderkongreß zur „CAT '87" angefordert werden. Die internationale Fachausstellung über computerunterstützte Technologien in der Fertigungsindustrie findet vom 02.06.87 bis 05.06.1987 statt.

Apple erfolgreich

Die Apple Computer Inc. hat im zweiten Quartal des Geschäftsjahres 1987 einen Umsatz von 575,3 Millionen Dollar erzielt. Das entspricht einer Steigerung um 41%. Der Reingewinn lag mit 33,9 Millionen Dollar 6% über dem des Vergleichszeitraumes. Die Ausgaben für Forschung und Entwicklung stiegen um 43% und ließen damit den Zuwachs beim Reingewinn im Vergleich zum Umsatz geringer ausfallen. Als Folge dieser Aufwendungen werden im Verlauf dieses Jahres weitere neue Produkte angekündigt. John Sculley, Chairman und Chief Executive Officer zeigte sich besonders zufrieden mit der hohen Akzeptanz des Macintosh SE. Verglichen mit früheren Macintosh-Produkteinführungen seien die hohen Auftragseingänge für den Macintosh SE einmalig in der Geschichte von Apple. Die Zuwachsraten der deutschen Apple Computer GmbH liegen nach Aussage des Geschäftsführers Gerhard Jörg deutlich über denen des Gesamtkonzerns. Desktop Publishing und die Einbindung in größere Netzwerke sollen hier ausschlaggebend sein.

Digital Research gegen Tatung

Digital Research hat die Software-Lizenz mit Tatung gekündigt und gerichtliche Schritte wegen vermutlicher Schutzrechtverletzung durch das unerlaubte Kopieren der Logo-Dokumentation von Digital Research eingeleitet. Das Unternehmen behauptet, daß Tatung ihren „Einstein-PC" mit Handbüchern ausgeliefert hat, die stellenweise direkt von der Digital Research-eigenen Dokumentation abkopiert wurden.

„Public-Domain"-Betriebssystem für den NDR-Klein-Computer

Auf dem NDR-Klein-Computer laufen heute mehr als ein Dutzend Betriebssysteme. Keines davon ist so verbreitet, daß man es bei fast allen Systemen voraussetzen kann. Dieser Zustand soll sich jetzt ändern. Mit der Vorstellung von System-DOS 5.0 hat das Software-Haus System-Soft nämlich die bisherige Version (4.x.x) als „Public-Domain"-Software freigegeben. Im Rahmen des Update-Service der Firma erhalten die registrierten Benutzer dieser Version das neue Betriebssystem einschließlich der neuen Handbücher kostenlos ausgetauscht.

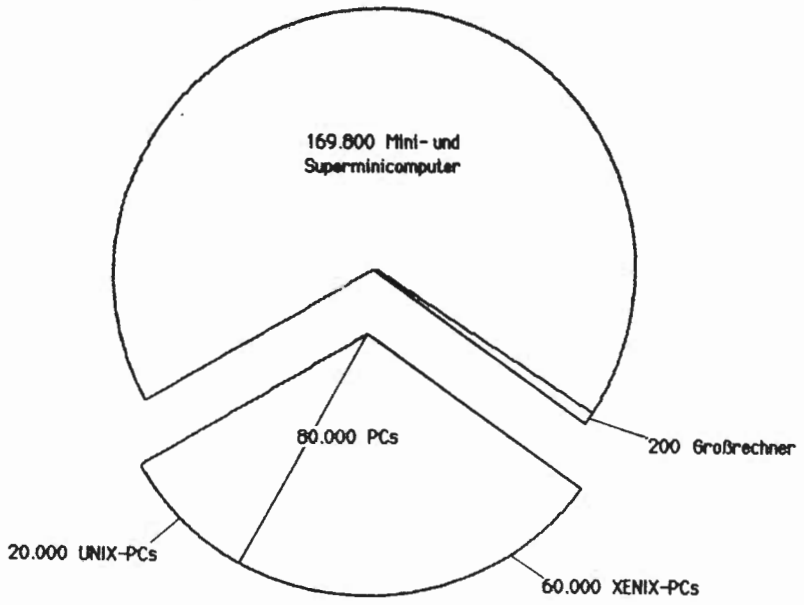
Die freigegebene DOS-Version läuft auf den CPUs 68008 und 68000. Sie setzt das Grundprogramm (Version 4.3) voraus. Das „freie" Betriebssystem kann vom Elektronikladen Detmold bezogen werden. Dafür fallen Kosten für das Kopieren und ein mitgeliefertes kleines Handbuch von 25 DM (5¼-Zoll-Format) bzw. 35 DM (3½-Zoll-Format) an.

Unix wird immer wichtiger

Im Jahre 1990 werden nach Schätzung des Eschborner Marktforschungsinstituts IDC weltweit über zwei Millionen Computer mit dem Betriebssystem Unix ausgestattet sein. Ausgehend von der gegenwärtig installierten Basis errechnet das Unternehmen eine durchschnittliche jährliche Zuwachsrate von 51% bis 1990. Dann werden PCs den weltweiten Unix-Markt mit rund 65% Anteil – das wären über eine halbe Million PCs – an installiertem Rechnerbestand dominieren.

UNIX : DURCHDRINGUNGSGRAD 1 BIS 10 PROZENT

Per Ende 1985 waren weltweit rund 250.000 UNIX-Computer (einschl. Derivate) installiert. Die Bestandsverteilung ist aus dem nachfolgenden Kreisdiagramm ersichtlich.



Anteile der UNIX-Rechner am jeweiligen Marktsegment (instal. Bestand):

Großrechner : 1,04 %
Minis/Superminis : 9,62 %
Mikrocomputer : 0,62 % (ohne Homecomputer)

So sieht IDC den Unix-Markt per Ende 1985

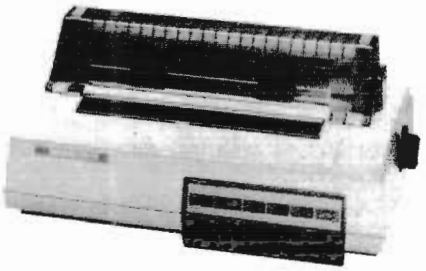
IDC CHART

NEC

Der Schnelle

NEC Pinwriter P9XL

- bis 400 Zeichen pro Sekunde – entspricht 6 DIN-A4-Seiten pro Minute
- 133 Zeichen pro Sekunde in Buchdruckqualität
- Grafikauflösung bis zu 360x360 Punkten/Zoll
- MTBF-Faktor von 7.000 Betriebsstunden (entspricht einer Druckerleistung von 4 Millionen Seiten)
- Große Schriftenvielfalt und Farbdruck



mc 6/87

Coupon

☐ Bitte senden Sie mir mehr Informationsmaterial
☐ Bitte senden Sie mir Ihre Firmenzeitschrift DISPLAY

Name _____

_____

Straße _____

PLZ/Ort _____

Tel.: _____

Computer-Peripherie-Geräte GmbH
6455 Erlensee · Beethovenstraße 26
Telefon 0 61 83/83-0

NEC fusioniert mit NEC

Die bisher getrennt operierenden Firmen NEC Home Electronics (Europe) GmbH, Neuß, und NEC Business Systems (Deutschland) GmbH, München, werden künftig ihre Aktivitäten gemeinsam betreiben. Das neue Unternehmen firmiert unter der Bezeichnung NEC Deutschland GmbH. Dieser Schritt ergibt sich aus den Bemühungen, den Verkaufs- und den Service-Bereich zu optimieren. Damit kann der Kunde jetzt aus einer Hand Drucker, Plattenlaufwerke, Fernkopierer, Computer, Monitore und Videorecorder beziehen.

Weiterhin selbständig wird die NEC Electronics (Europe) GmbH, Düsseldorf, mit ihrem Tochterunternehmen NEC Electronics (Germany) GmbH, München, ihren Geschäften nachgehen. Diese Firma ist für elektronische Bauelemente zuständig.

Nachträge

Z80-Karte für den mc-68000-Computer

mc 1987, Heft 5, Seite 119

In Bild 4 (Teilschaltplan Host-Interface) fehlt am untersten Ausgang auf der rechten Seite (IC 30, Pin 11) der Signalname $\overline{RD}$ (aktiv low). Auch der Ausgang BUSRQ (rechts ganz oben) sollte eine Linie über dem Signalnamen haben, denn auch er ist aktiv low.

Grafik fertig, was nun?

mc 1986, Heft 10, Seite 58

In dem Programm befinden sich ein Fehler und einige Umständlichkeiten, die einem C-Anfänger das Leben schon schwer machen können. So werden die Dateien vom Programm mit
stream = fopen (name, access);
geöffnet. Dies ist die richtige Methode für ASCII-Dateien; die bei der Übertragung von Bilddaten auftretenden Nicht-ASCII-Bytes können hier jedoch zu Übertragungsfehlern führen (und werden das in der Regel auch tun). Die Datei muß also als Binär-Datei geöffnet werden, d. h. mit
stream = fopenb (name, access);
oder, beim Anlegen einer noch nicht existierenden Datei, mit

fd = creatb (name, mode);
stream = fdopen (fd, access);

Erklärungen der hier verwendeten Parameter gibt „The C Language Programming Guide for CP/M-68k" von Digital Research.
Die Umständlichkeit besteht darin, daß die Bilddaten innerhalb einer Schleife und mit aufwendigen Manipulationen übertragen werden. C bietet jedoch die Möglichkeit, mit einem Befehl eine beliebige Anzahl Bytes auf der Diskette abzuspeichern:
[nitems =] fread (buff, size, nitems, stream);
bzw.
[nitems =] fwrite (buff, size, nitems, stream);

Außerdem wird man ein Bild selten aus der Betriebssystemebene heraus abspeichern oder laden wollen, sondern meist aus einem Programm heraus. Wenn man die Routinen jedoch als Unterprogramme schreibt, erübrigt sich die Verwendung von argc und argv. Die Unterprogramme zum Speichern und Lesen, zusammen mit einem Rumpfprogramm, zeigt das *Bild*.

Roland Winde

```
#include <stdio.h>

main()
{
char  *name;

  strcpy (name,"test.pic");              /* Filename              */
  savepic(name);                         /* Bild speichern        */
  printf ("%c",12);                      /* Bildschirm loeschen   */
  loadpic (name);                        /* Bild wieder laden     */
}


savepic(name)
char *name;
{
FILE   *fdopen(), *fopenb(), *stream;
long   vram;
short  fd;

  vram = 0xFF8000;              /* Basis-Adresse (bis Monitor 1.4) */
  vram = (*(long*)0xFFE9B0);    /* Basis-Adresse  (ab Monitor 1.4) */

  if (access(name,4))                   /* existiert File ?        */
     {
     fd = creatb (name,0);              /* Nein, neu anlegen       */
     stream = fdopen (fd,"w");
     }
  else
     stream = fopenb (name,"w");        /* Ja, ueberschreiben      */

  fwrite (vram,0x4000,1,stream);        /* 16kB Bilddaten schreiben */
  fclose (stream);                      /* File schliessen         */
}


loadpic(name)
char *name;
{
FILE *fopenb(), *stream;
long vram;

  vram = 0xFF8000;              /* Basis-Adresse (bis Monitor 1.4) */
  vram = (*(long*)0xFFE9B0);    /* Basis-Adresse  (ab Monitor 1.4) */

  if (access(name,4))                   /* File vorhanden ?        */
     printf ("File not found\n");       /* Nein                    */
  else
     {                                  /* Ja                      */
     stream = fopenb (name,"r");        /* oeffne Binaer-File      */
     fread (vram,0x4000,1,stream);      /* lies 16kB Bilddaten     */
     fclose (stream);                   /* schliesse File          */
     }
}
```

Die beiden Unterprogramme savepic und loadpic lassen sich leicht in Grafik-Programme einbinden

mc-Sonderheft „Turbo-Pascal im Einsatz“

Turbo-Pascal hat sich, nicht zuletzt wegen des günstigen Preises, als Standard für die Programmiersprache Pascal durchgesetzt. Das Sonderheft „Turbo-Pascal im Einsatz“ wendet sich an alle, die mit dieser Sprache arbeiten, wobei der Schwerpunkt im Bereich der MS-DOS-Computer liegt. Hier einige Themen des interessanten Sonderheftes: Ansteuern der Hercules-Karte, Druckergrafik mit Matrixdruckern, Platten- und Dateiverwaltung, Programmieren von EPROMS am Druckerausgang, Programmsteuerung mit der Microsoft-Maus, Abfangen von Fehlern durch Umleiten der Interrupts, Programmierung von quasi parallel ablaufenden Tasks, Einbinden von Systemaufrufen. Für den Apple-II wurde die Grafikausgabe in das Sonderheft aufgenommen. Viele Tips, Tricks und Kniffe werden verraten, z. B. eine Routine zum Rückverfolgen von Fehlern. Viele bewährte Beiträge wurden aus der mc übernommen, etwa 30 % der Beiträge sind neu. Das Sonderheft der mc „Turbo-Pascal im Einsatz“ erscheint am 12.06.87, hat 112 Seiten und kostet 24 DM. Man bekommt es bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, in Mikrocomputer-Shops, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder direkt beim Franzis-Verlag.

Zenith liefert Z-386 aus

Der auf der CeBIT im März erstmals vorgestellte 80386-PC von Zenith wird jetzt ausgeliefert. Das Einstiegsmodell mit einer 40-MByte-Festplatte ist für 18 000 DM zu haben. Serienmäßig haben die 386er, die es auch mit einer 80-MByte-Festplatte gibt, 1 MByte RAM ohne Wartezyklen, ein Diskettenlaufwerk mit 1,2 MByte und fünf freie Steckplätze für 8-, 16- und 32-Bit-Steckkarten.
Die Entwickler haben bei der Systemarchitektur besonderen Wert auf schnellste Speicheroperationen gelegt. Ein Enhanced Page-Mode RAM erlaubt, Blöcke von 2 KByte oder 4 KByte mit einem Adressierungsvorgang zu schreiben oder zu lesen. Durch einen speziellen Page-Mode-Controller, der in die Systemarchitektur integriert ist, soll der Transfer dieser Blöcke um weitere 40 % gesteigert werden. Beim Speicher-Refresh werden im Z-386 vier

Das Flagschiff von Zenith, der Z-386, ist durch spezielle Systemarchitekturen besonders schnell

Reihen Speicher auf einmal bearbeitet. Wem die gebotene Geschwindigkeit nicht langt, kann ein spezielles Cache-Memory mit einer Zugriffszeit von 35 ns nachrüsten.
Betriebssystem-Routinen werden teilweise in besonders schnelle RAMs kopiert, von wo aus sie bedeutend schneller als von den ROMs aus agieren. Der Zenith Z-386 wird mit MS-DOS 3.2 und MS-Windows geliefert.

Bringt den PC auf Touren

Eine Steckkarte von Verbatim mit der Bezeichnung DataLife macht aus einem normalen IBM-PC einen AT. Sie enthält neben der CPU 80286 auch eine Fassung für den Arithmetik-Coprozessor 80287. Je nachdem, welche Version man davon einsetzt, läßt sich der Zusatz-Chip mit 7,2 MHz oder 4,8 MHz Taktfrequenz betreiben. Die Verbindung zum PC bildet ein Bandkabel, das zur Fassung der bisherigen CPU führt. Deren Speicher- und E/A-Zugriffe werden von der neuen Hardware simuliert. Diese Methode ist bei derartigen Beschleunigerkarten allgemein üblich. Probleme mit IBM-kompatiblen Systemerweiterungen, Peripheriegeräten und Programmen sind nicht zu erwarten. Wer nicht den Original-PC, sondern ein dazu kompatibles Gerät umbauen will, sollte darauf achten, daß hier ausnahmsweise nicht allein die Kompatibilität der Elektronik wichtig ist. Es ist nämlich nicht ausgeschlossen, daß ganz einfach das Bandkabel zu kurz ist, um die Strecke von der 8088-Fassung zum Steckplatz zu überbrücken.

NEC

Der Vielseitige

NEC Pinwriter P5XL Koax

- Schnelldruck – 4 Seiten pro Minute
- Korrespondenzqualität – 88 Zeichen pro Sekunde
- Carbonfarbband für Textverarbeitung in Buchdruckqualität
- Brillanter Farbdruck in 7 Farben
- Grafikauflösung bis zu 360x360 dpi
- Anschlußmöglichkeit an IBM Großrechenanlagen
- 6 Schriften serienmäßig, viele zusätzliche (auch OCR-A/B) durch Fontcardriges
- Verschiedene Papierzuführungsoptionen

mc 6/87

Coupon

☐ Bitte senden Sie mir mehr Informationsmaterial
☐ Bitte senden Sie mir Ihre Firmenzeitschrift DISPLAY

Name _____

_____

Straße _____

PLZ/Ort _____

Tel.: _____

Computer-Peripherie-Geräte GmbH
6455 Erlensee · Beethovenstraße 26
Telefon 0 61 83/83-0

Zenith senkt Preise

Für weniger als 2000 DM bekommt man jetzt einen vollwertigen PC mit zwei Diskettenlaufwerken, den Zenith College PC Z-148. Der Preis für den Computer, der mit 512 KByte RAM, Color-Grafikkarte, Parallel- und Seriellschnittstelle und einem monochromen Monitor geliefert wird, wurde damit um 500 DM gesenkt. Gleich um 1000 DM billiger ist die Version des Z-148 College PCs mit 20-MByte-Festplatte geworden, die jetzt nur noch 2999 DM kostet.
Zenith begründet die Preissenkungen mit dem erneut gesunkenen Dollarkurs und als Reaktion auf eine Preissenkungswelle in den USA. Im vergangenen Jahr sollen mehr als 10 000 College PCs abgesetzt worden sein, womit sich das Unternehmen nach Ansicht des Geschäftsführers Werner Reinkendorff im Marktsegment der preiswerten PCs eine führende Position erobert hat, die jederzeit verteidigt werden soll.

Der Z-148 College PC von Zenith ist für 1999 DM mit zwei Diskettenlaufwerken voll einsetzbar. Mit Festplatte sind 2999 DM zu zahlen.

Pascal nach ISO

Seit kurzem bietet die AC Copy GmbH aus Aachen mit Pascal-II einen MS-DOS-Pascal-Compiler an, der sich voll an den ISO-Standard für diese Programmiersprache hält. Das ist besonders dann interessant, wenn Pascal-Programme auf dem PC und auf Rechnern aus dem Mini- und Mainframebereich laufen sollen. Das Pascal-II kommt von Oregon Software und nutzt den gesamten Speicher. Es erzeugt Object-Files, die mit dem MS-DOS-Linker zusammengebunden werden können. Eine integrierte Entwicklungsumgebung aus Editor, Compiler und Programmtest im Speicher gibt es hier nicht. Zu dem System, das auf der Festplatte installiert werden muß (mindestens 1,5 MByte müßen frei sein), bekommt man einige Tools dazu. Ein automatischer Formatierer bringt den Quelltext in ansprechende Form, mit einem Cross-Referenz-Programm behält man Übersicht auch über große Programme, ein Debugger dient der Fehlersuche in compilierten Programmen und zahlreiche Makros vereinfachen das Einbinden von Assembler-Routinen. Vom Compiler werden die Prozessoren 80286 und 8087 unterstützt. Die Umstellung von Turbo-Pascal nach Pascal-II wird durch ein Hilfsprogramm vereinfacht.

Eumel für Compaq Deskpro 386

Das Mannheimer Softwarehaus Webidata gab die Anpassung des Multiuser-/Multitasking-Betriebssystems Eumel an den Compaq Deskpro 386 bekannt. Bis zu 15 Terminals können mit dem 1600 DM teueren System, zu dem ein Elan-Compiler und eine Textverarbeitung dazugehören, benutzt werden.

Fundgrube für PC-Fans

„Das PC-Sonderheft", das jetzt im Franzis-Verlag erschienen ist, erweist sich als wahre Fundgrube für alle, die mit dem IBM-PC und dazu kompatiblen Computern arbeiten. Auf insgesamt 96 Seiten gibt es dem Leser Tips und Hintergrundinformationen zum Innenleben des PC, präsentiert Grafikprogramme, Software für die Anpassung von Druckern und zahlreiche Utilities.
Mehrere Beiträge gehen auf die Struktur und die Anwendung der EGA-Karte ein. Der Leser findet zahlreiche Turbo-Pascal-Routinen, mit denen er diese Karte ansprechen kann. Ähnliche Informationen gibt es zur weitverbreiteten Hercules-Grafikkarte.
Das neue Sonderheft faßt zahlreiche Beiträge aus der mc zusammen, enthält aber auch eine Menge Neues. Es kostet 24,50 DM und ist direkt beim Franzis-Verlag, im Buch- oder Zeitschriftenhandel sowie in Elektronik- oder Mikrocomputer-Shops erhältlich.

TRANSPUTER-SYSTEME

Parallele Rechnerarchitekturen bringen die Leistung von Supercomputern auch auf Ihren Schreibtisch.
Jetzt ist die erforderliche Hard- und Software lieferbar.

Transputer-Karte TRI

Bestückt mit dem Inmos Transputer T414 und 1 bzw. 2 MByte RAM macht dieses ADD-ON-Board Ihren PC zum „Personal Supercomputer".

Transputer-Software-Entwicklungsumgebung

Symbolischer 2-Pass-Assembler, Disassembler, TDS/MS-DOS-Converter, Ladeprogramm und Software-Interface zu IBM PC XT/AT bieten jetzt eine Entwicklungsumgebung, die Ihnen alle Tools für die Programmierung von Transputer-Applikationen in die Hand gibt

Anwendungen der Artificial Intelligence, Bildverarbeitung, Signalanalyse, Klassifikation, etc. sind mit dem Transputer aufgrund der enormen Rechenleistungen kein Problem mehr.

Fordern Sie ausführliche Informationen an:

Brainware
Ihr Experte in Expertensystemen
Consulting · Schulung · Software

Brainware GmbH · Gustav-Meyer-Allee 25
1000 Berlin 65 · Tel. 030/469 46 96 oder
Tel. 030/463 30 48 · Teletex 308 145-BIG

Praxis des Multitasking

Von Michael Zilker. 112 Seiten mit 34 Abbildungen. 38 DM. Lwstr-kart. Franzis-Verlag, München.
ISBN 3-7723-8561-3

Bekannt sind Multitasking-Betriebssysteme wie Unix und OS-9. Doch wie sieht es hinter den Kulissen eines Multitasking-Betriebssystems aus?
Der Autor zerlegt das komplexe Thema in die Bereiche Betriebssystemkern, Semaphoren, Datentransport zwischen Tasks, Interrupt- und Speicherverwaltung. Damit das Ganze nicht pure Theorie bleibt, ist das Listing des vom Autor entwickelten Betriebssystems BS0 komplett abgedruckt. BS0 benötigt als Prozessor den weitverbreiteten Z80 und ist nur ungefähr 3 KByte groß. Der Leser kann das in den verschiedenen Kapiteln erworbene Wissen anhand des gut kommentierten Listings vertiefen. Verständlich dargestellt sind die Aufgaben der beiden Schlüsselelemente eines Multitasking-Betriebssystems – der Dispatcher und der Scheduler. Verschiedene Scheduling-Verfahren (unterbrechende Steuerung und Round-Robin-Verfahren) und ihr Anwendungsbereich werden skizziert. Sehr gut gefällt mir die Darstellung der klassischen Problematik eines Multitasking-Betriebssystems: die Konkurrenz um Betriebsmittel. Ausgehend von dem Konzept der Synchronisation von Prozessen durch Flags geht der Autor auf das erstmalig von E. W. Dijkstra entwickelte Semaphorkonzept ein. Die Wirkung der grundlegenden Semaphoroperationen Signal und Wait wird ausführlich beschrieben.
Das Buch ist leicht verständlich und für jeden geeignet, der Grundkenntnisse in der Mikroprozessortechnik hat.

St

Optische Speicher im Überblick

Optische Speicher. Von Jürgen Schulte-Hillen und Ulrich Schwerhoff. 122 Seiten. 68 DM. Klaes Verlag, Essen 1986.
ISBN 3-925506-05-5

Als ein akustische Signale speicherndes Mitglied der Familie der Compact-Discs gehört die CD-Audio mittlerweile zum Musik-Alltag. Ungewohnt dagegen ist eine CD-ROM, jene Art von Datenspeicher, dessen Aufnahmefähigkeit Magnetband und Mikrofilm erblassen läßt.
Dieses Buch ist für jene gedacht, die es nicht bei der Faszination allein belassen, sondern sich näher mit der zukunftsweisenden Technik und den Anwendungsbereichen von optischen Massenspeichern befassen wollen.
Das Buch unterscheidet je nach speziellem Speicher- und Aufzeichnungsverfahren zwischen vier Gruppen. Die den Varianten Read-Only-, Write-Once-, Erasable-Speicher und einigen Sonderformen zugrundeliegenden Techniken zeigen die Autoren übersichtlich auf. Skizzen erleichtern das Verständnis.
Aufbauend auf dieser Grundlage werden die typischen Vertreter der verschiedenen Technologien vorgestellt.
Angesichts der derzeit herausragenden Bedeutung der CD-ROM richtet der zweite Teil des Buches das Augenmerk des Lesers auf diese Variante eines Read-Only-Speichers. Die Umstellung einer Datenbank auf CD-ROM inklusive einer Beschreibung der notwendigen Arbeitsschritte setzt den Schlußpunkt. Eine Liste der gebräuchlichen Begriffe und Produkte der optischen Massenspeicherung rundet das Buch ab. Lediglich der Preis des Paperbacks wird an dem einen oder anderen Geldbeutel anecken.

Monika Kagerbauer

Berufsberater

Arbeitsfeld Computer. Von Kurt Weichler. 268 Seiten. 12.80 DM. Rohwohlt Verlag, Reinbek 1987.
ISBN 3-499-18702-7

Gleich auf den ersten Seiten seines Ratgebers hilft Kurt Weichler durch die Beschreibung individueller Karriereleitern – die an den amerikanischen Traum vom Tellerwäscher zum Millionär erinnern, mit, dem an Computer-Berufen Interessierten dieses Betätigungsfeld schmackhaft zu machen.
Doch ausschließlich im ersten Augenblick wird der Leser geblendet von solch rosigen Situationsaufnahmen in der Computerbranche. Amüsant verpackte Hinweise darauf, daß nicht jeder für einen Computer-Beruf geeignet ist können vor einer überstürzten Berufswahl bewahren.
Wer nach wie vor mit dem Gedanken spielt, dem werden auf rund 80 Seiten „Computerberufe im Profil" vorgestellt. Eine detaillierte Aufstellung der einzelnen Arbeitsfelder strukturiert die nicht nur für den Laien oft verwirrende Vielzahl der Berufsbezeichnungen mit den dazugehörigen Abschlüssen.
Damit die Finanzierung einer EDV-Ausbildung nicht zum Problem wird, informiert der Autor über Förderungsmöglichkeiten.
Für denjenigen, der bereits in der Ausbildung steht oder diese gar schon hinter sich hat, gibt es Tips, wie man am elegantesten mit potentiellen Arbeitgebern in Kontakt kommt. Indem es bei der Beurteilung neuer Techniken einseitige Betrachtungsmuster vermeidet, liefert das Buch wertvolle Anhaltspunkte. Die letztendliche Entscheidung kann es freilich dem unschlüssig an seiner beruflichen Zukunft Bastelnden nicht abnehmen!

Monika Kagerbauer

Einführung in Turbo-Pascal

Programmieren mit Turbo-Pascal 3.0. Von Winfried Kassera. 316 Seiten. 59 DM. Markt & Technik-Verlag, Haar bei München 1986.
ISBN 3-89090-159-X

Winfried Kassera widmet sich dem Kreis der Turbo-Pascal-Anhänger, wobei sein Buch als praktische Anleitung zur Arbeit mit diesem Pascal-Dialekt gedacht ist. Anhand zunächst einfacher, bald komplexerer Beispiele erläutert der Autor die Turbo-Pascal-Syntax. Fast 100 Beispiele – die auf einer dem Buch beiliegenden Diskette enthalten sind – wurden gründlich aufbereitet.
Da bald schon strukturierte Datentypen wie STRING und ARRAY benutzt werden ist ein wenig Vorwissen über Programmiersprachen sicherlich von Vorteil. Am Beispiel des Zauberwürfels wird ein umfassendes Programm mit Menütechnik, Zeigervariablen und Bildschirmadressierung aufgebaut. Verfeinerten Arbeitstechniken, wie der Erstellung von Bildschirmmasken widmet Kassera ein eigenes Kapitel, ebenso der in Turbo-Pascal günstig gelösten Form der Dateiarbeit.
Dem Fortgeschrittenen bietet der Autor schließlich Anregungen für den Umgang mit Maschinenbefehlen und Systemfunktionen.
Im Anhang finden sich Übersichten über Prozeduren/Funktionen, Compiler-Direktiven und über die Editor-Bedienung.

Da es immer etwas problematisch ist, eine Sprache ausschließlich anhand zahlreicher Beispiele zu erklären – was eine Behandlung der vollständigen Strukturen an Ort und Stelle ausschließt – würde sich in einer weiteren Auflage ein Anhang mit der gesamten Syntax empfehlen.

Dr. Thilo Weller

Holger Haase, Michael Menrad

mc-AllProg

PALs, EPROMs, PROMs ... auf einem Gerät programmieren, Teil 1

Wer ICs programmieren will, stolpert beim Studieren der Datenblätter sehr schnell über viele Schwierigkeiten. Das beginnt bei den benötigten Spannungen, die meist jenseits der freundlichen 5 Volt liegen. Dann sind Programmierpulse mit definierten Zeiten nötig. Auch die Vorschriften für die logischen Pegel sind meist strenger als beim späteren Einsatz der Bausteine. Weiter kommen oft noch eine Programmier-Leistungsaufnahme von einigen Watt und die unterschiedlichsten Flankensteilheiten der Programmierimpulse bei den verschiedenen IC-Familien und -Herstellern dazu. Um all diesen Forderungen gerecht zu werden, wurde hier auf einer Doppel-Europakarte eine Schaltung um den Prozessor SAB 80535 aufgebaut. Diese Karte, mit ihrer seriellen Schnittstelle zur Außenwelt, kann fast alle ICs programmieren, die im täglichen Leben eines Professionals vorkommen.

Man möchte meinen, daß die Firmen, die programmierbare ICs herstellen, seien es PALs, EPROMs oder andere Typen, sich freuen, wenn jemand ein Gerät baut das die ICs im Firmenprogramm programmieren kann. Wir mußten bei der Arbeit an unserem universellen Programmiergerät (*Bild 1*) feststellen, daß dies nicht immer der Fall ist. Oft haben wir nur unter großen Schwierigkeiten die notwendigen Programmier-Verfahren herausfinden können. Es mag daran liegen, daß die Halbleiterfirmen bisher nur mit Großkunden aus der Industrie zu tun hatten und deshalb Anfragen einzelner Personen nur ungern beantworten. Wir sind nun stolz, daß wir hier in den *Bildern 2* bis *4* zeigen können, daß mit dem Programmiergerät mc-Allprog wirklich fast alles „Brennbare" programmiert werden kann.

Diese Vielfalt von IC-Typen kann man nur dann auf einem Gerät programmieren, wenn es universell einstellbar ist,

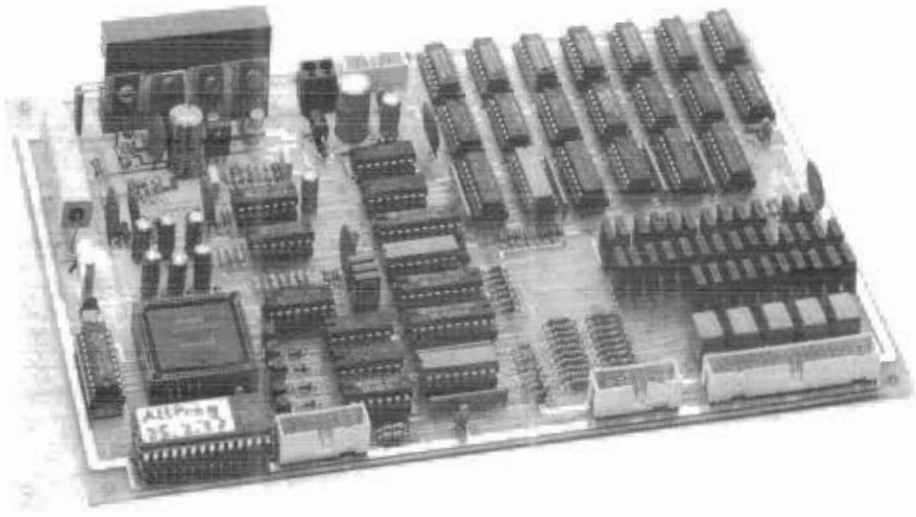


Bild 1. Die bestückte Platine

Name	Progr. Spann.	Organisation	Progr modi
2516	25.0 V	2Kx8	N
2716	25.0 V	2Kx8	N
2532	25.0 V	4Kx8	N
2732	25.0 V	4Kx8	N
2732A	21.0 V	4Kx8	N
2764	21.0 V	8Kx8	N S
2764	12.5 V	8Kx8	N S
27128	21.0 V	16Kx8	N S
27128	12.5 V	16Kx8	N S
27256	21.0 V	32Kx8	I Q
27256	12.5 V	32Kx8	I Q
27512	21.0 V	64Kx8	I Q
27512	12.5 V	64Kx8	I Q
27513	12.5 V	4x16Kx8	I Q
27010	12.5 V	128Kx8	I Q
27011	12.5 V	8x16Kx8	I Q

Programmiermodi:
N - Normal
S - Schneller interaktiver Modus
I - INTELligent Modus
Q - Quick Modus

Bild 2. Diese EPROMs können programmiert werden

Typ	Organisation	Ausgang
82S23	32x8	OC
82S23A	32x8	OC
82S123	32x8	TS
82S123A	32x8	TS
82S126	256x4	OC
82S126A	256x4	OC
82S129	256x4	TS
82S129A	256x4	TS
82S130	512x4	OC
82S130A	512x4	OC
82S131	512x4	TS
82S131A	512x4	TS
82LS135	256x8	TS
82S135	256x8	TS
82S115	512x8	TS
82S141	512x8	TS
82S137	1024x4	TS
82S137A	1024x4	TS
82S137B	1024x4	TS
82S147	512x8	TS
82S147A	512x8	TS
82LS147	512x8	TS
82LS181	1024x8	TS
82S180	1024x8	OC
82S181	1024x8	TS
82S181A	1024x8	TS
82S181B	1024x8	TS
82S183	1024x8	TS
82S185	2048x4	TS
82S185A	2048x4	TS
82S185B	2048x4	TS
82S191	2048x8	TS
82S191A	2048x8	TS
82S191B	2048x8	TS
82S321	4096x8	TS

Ausgang :
OC => Open Collector
TS => Tri-State

Bild 3. Das sind die PROMs, die mc-Allprog bearbeiten kann

mit anderen Worten: wenn es selbst programmierbar ist. Zu mc-Allprog wird also ein Software-Paket gehören, das auf einem CP/M-80-Rechner oder einem MS-DOS-Recher läuft. In diesem Paket wird unter anderem ein PAL-Assembler eingeschlossen sein, denn diese IC-Sorte kann man nicht mehr von Hand bearbeiten. Es wird sogar ein Simulations-Teil im Programmpaket enthalten sein, mit dem man zunächst testen kann, ob man richtig programmiert hat, ohne gleich teure PAL-Bausteine zu zerschießen. Die Platine, deren Aufbau in diesem ersten Teil geschildert wird, besitzt eigenständige Intelligenz und wird über eine serielle Schnittstelle mit Daten vom Software-Paket versorgt.

Die Platine benötigt viele Spannungen

Die Schaltung benötigt verschiedene Spannungen. Für die Versorgung der RS-232-Treiber und der Operationsverstärker sind je eine positive und eine negative Spannungsquelle nötig. Diese Spannungen sollten 8 bis 15 Volt betragen. Sie müssen stabilisiert sein. Dazu kommt die Energieversorgung der TTL Bausteine. Die 5 Volt für diese Bausteine werden auf der Platine geregelt. Es müssen dafür also mindestens 7,5 Volt bereitgestellt werden. Bei den meisten Computern kann man diese drei Spannungen (*Tabelle 1*) aus dem vorhandenen Netzteil abzweigen. Deshalb sind sie zusammen mit den Signalen für die serielle Verbindung über den Stecker ST1 geführt (*Bild 5*).

Weiter benötigt die Platine die Energieversorgung für die Programmierung der ICs (*Bild 6*). Diese Spannung wird über die Klemme KL zugeführt. Sie muß zwischen 18 Volt und 35 Volt liegen und mit Kondensatoren geglättet werden. Diese Spannung muß außerdem mindestens 2,5 Volt über der benötigten Programmierspannung liegen. Sie und der dazu gehörende Strombedarf richten sich fast ausschließlich nach den zu programmierenden Bausteinen. So käme man, sofern man auf die Programmierung einiger IC-Typen verzichten will, mit einem kleinen Netzteil oder einem DC/DC-Wandler aus. Dazu zwei Beispiele: Um EPROMs zu programmieren, reichen oft 18 Volt und 100 mA aus. Bei einigen PROMs muß die externe Spannung dagegen über 32 Volt liegen und mit 1,5 Ampere belastbar sein. Einen Überblick über diese Spannungen und Ströme gibt *Tabelle 2*.

Tabelle 1: Die Spannungen, die vom Computer kommen (Pfostenstecker St 1)

Spannung		Strom	Bemerkungen
VP	+7,5 V	1 A	R1 = 0 Ω
	+9 V	1 A	R1 = 2,7 Ω (5 W)
	+12 V	1 A	R1 = 4,7 Ω (7 W)
	+15 V	1 A	R1 = 6,8 Ω (9 W)
V24P	+8...+15 V	100 mA	stabilisiert
V24N	−8...−15 V	50 mA	stabilisiert

Tabelle 2: Diese Spannungen werden für KL benötigt

IC-Familie	min. Spannung	max. Strom
EPROM (12,5 V)	18 V	80 mA
EPROM (21 V)	24 V	100 mA
PAL	24 V	1 A
EPROM (24 V)	28 V	100 mA
PROM (TiW-Bits)	18 V	1,5 A
PROM (NiCr-Bits)	32 V	800 mA

Tabelle 3: Die Baudrateneinstellung

Übertragungsrate	J2	J1
9600 Bd	offen	offen
4800 Bd	offen	geschlossen
2400 Bd	geschlossen	offen
19200 Bd	geschlossen	geschlossen

Die serielle Schnittstelle

Die serielle Verbindung zum steuernden Rechner arbeitet mit 8 Datenbit, einem Startbit und einem Stopbit. Die Baudrate läßt sich über die Jumper J1 und J2 einstellen (*Tabelle 3*). Für den Handshake-Betrieb sind die Signale DTR und CTS vorgesehen. Werden sie nicht benötigt, so kann man CTS mit V24P verbinden und DTR offen lassen. Sonst muß man das Signal DTR der Programmierplatine mit CTS am Hauptrechner verbinden und umgekehrt. Weitere, nicht benutzte Eingangssignale dieses Rechners sind auf Zustände zu legen, die die Schnittstelle nicht blockieren.

Das Handshaking funktioniert nun folgendermaßen: Ist das DTR-Signal aktiv (positive Spannung), so darf gesendet werden. Der Sender von DTR zeigt so seine Empfangsbereitschaft. Diese wird durch den CTS-Eingang von der Gegenstelle vor dem Senden eines jeden Bytes überprüft. Bei einer Z80-SIO muß man den DTR-Ausgang vom Programm her bedienen, während CTS von der SIO automatisch interpretiert werden kann. Ist diese DTR-Software auf dem Hauptrechner nicht vorhanden, so funktioniert das Handshaking nur in einer Richtung.

PAL-Typ	PAL-Hersteller MMI	TI	NSC	AMD
10H8	X		X	
12H6	X		X	
14H4	X		X	
16H2	X		X	
16C1	X		X	
10L8	X		X	
12L6	X		X	
14L4	X		X	
16L2	X		X	
16L8	X	X	X	X
16H8				X
16LD8				X
16HD8				X
16R8	X	X	X	X
16R6	X	X	X	X
16R4	X	X	X	X
16X4	X		X	
16A4	X		X	
12L10	X		X	
14L8	X		X	
16L6	X		X	
18L4	X		X	
20L2	X		X	
20C1	X		X	
20L10	X		X	
20X10	X		X	
20X8	X		X	
20X4	X		X	
20L8	X	X		
20R8	X	X		
20R6	X	X		
20R4	X	X		

Bild 4. Die PALs, die programmiert werden können. Softwareunterstützung enthält das Programmpaket zu Allprog

Gut kühlen

Es ist wichtig, für eine gute Belüftung des Kühlbleches und des Widerstandes R1 (Bild 6) zu sorgen. Über die Temperatur des Kühlbleches wacht ein NTC

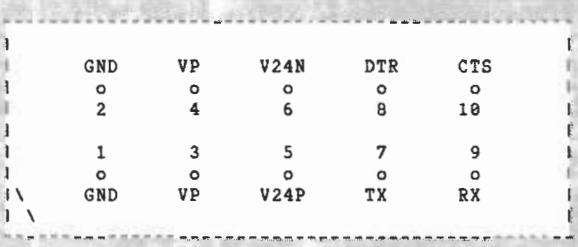


Bild 5. An Stecker 1 liegen die RS-232C-Signale und einige Spannungen

(R37). So kann bei Überhitzung eine Meldung an den angeschlossenen Rechner gegeben werden.

Der Prozessor

In der Schaltung von mc-Allprog arbeitet ein SAB 80535 (*Bild 7*) von Siemens. Seine Maschinensprache ist identisch mit der der Intel-Familie 8031/32. In dem 68poligen PLCC-Gehäuse dieses Ein-Chip-Rechners befinden sich 256 Byte RAM, acht Analogeingänge mit einem Multiplexer und einem AD-Wandler, drei Zeitgeber/Zähler und eine asynchrone serielle Schnittstelle. Weiter sind in der hier verwendeten Konfiguration noch 20 Portbits vorhanden. Sie sind mit Px.y als Port x, Bit y gekennzeichnet. Mit ihnen werden zum Beispiel die Signale CTS und DTR für die serielle Schnittstelle bedient. So sind zwischen dem Prozessor und dem Interface-Stecker ST1 nur noch Treiber bzw. Empfänger für die RS-232-Signale nötig.

Die Stromversorgung

Von den oben schon beschriebenen Spannungen werden aus VP in IC15 für den Betrieb des Prozessors und der TTL-ICs 5 Volt gemacht (Bild 6). Weil diese +5 Volt auf der mc-Allprog-Platine gleichzeitig Referenzspannung für die AD-Wandler im Prozessor sind, kann man sie nicht einfach aus dem Hauptrechner abzweigen (die notwendige Genauigkeit wäre dann, wenn überhaupt, nur mit einem sehr kurzen Verbindungskabel erreichbar). Da es sich beim 78S05 (IC15) um einen Längsregler handelt, entsteht leider viel Wärme, die durch den Widerstand R1 teilweise abgefangen werden soll. Seinen Wert muß man nach Tabelle 1 bestimmen.

Die DA-Wandler

Die Einstellung der verschiedenen Programmierspannungen übernehmen DA-Wandler. Da durch sie nur Gleichspannungen eingestellt werden müssen, also keine schnellen Änderungen erzeugt werden müssen, sind diese Wandler sehr einfach aufgebaut: Mit dem Zeitgeber 2 aus dem SAB 80535 und drei Vergleichsregistern (IC1) wurden die 14-Hz-

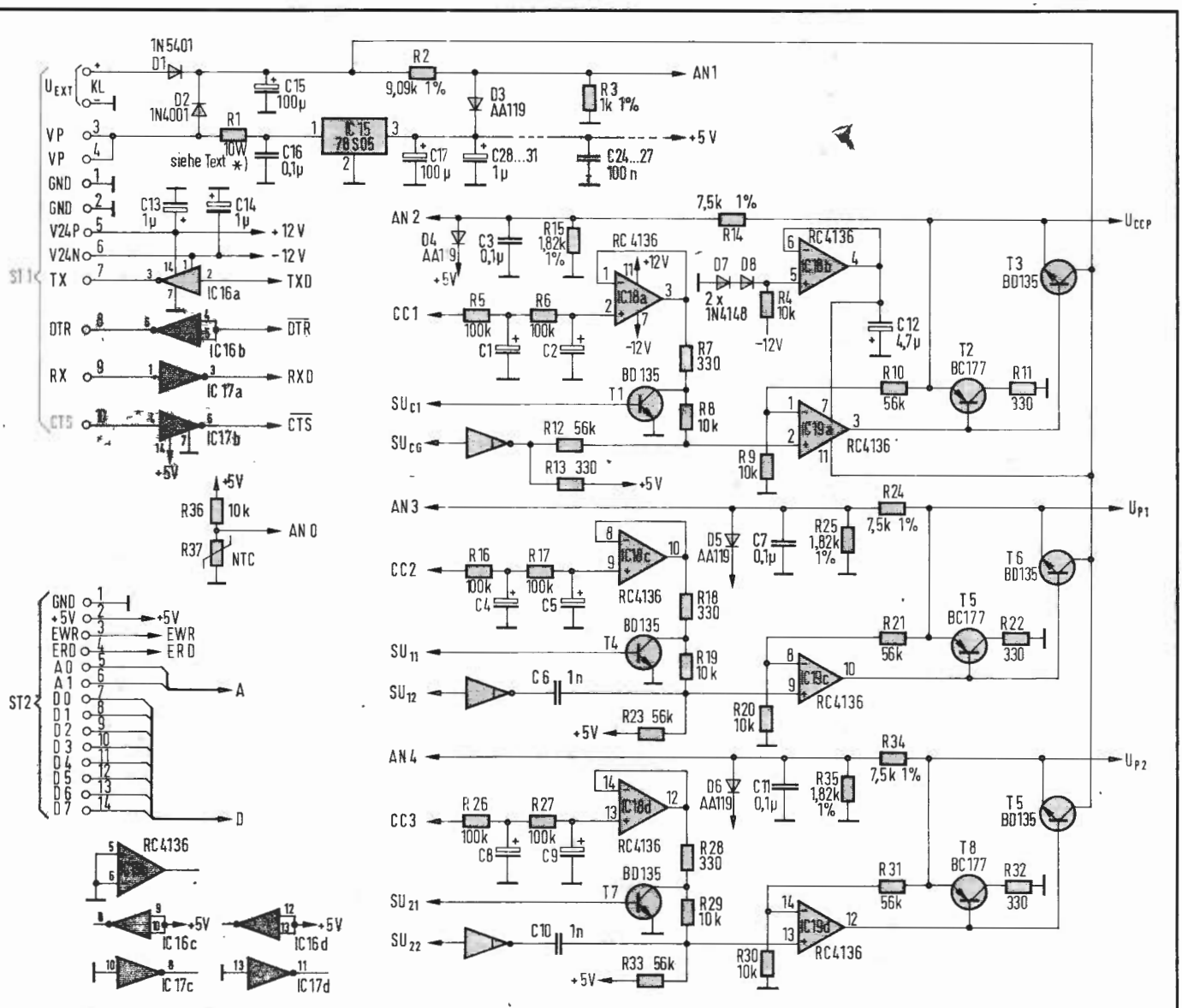


Bild 6. Der Analogteil von mc-Allprog. An die Klemme KL wird die über die Transistoren T3, T6 und T9 einstellbare Programmierspannungsversorgung angeschlossen

Ausgänge CC1 bis CC3 realisiert. Ihre Tastverhältnisse lassen sich mit einer Auflösung von 16 Bit einstellen. Durch nachgeschaltete Tiefpässe entsteht je eine zum Tastverhältnis proportionale Spannung. Um die Linearität der Anordnung zu verbessern, folgen je ein Impedanzwandler. Der zweite Operationsverstärker steuert jeweils die zwei Leistungstransistoren an (BD 135 gekühlt und BC 177). Hier werden die Signale um den Faktor 6,6 verstärkt. Aus den höchstens 4,5 Volt hinter dem Tiefpaß werden auf diese Weise 29,7 Volt. Die aus dieser Regelschaltung kommende Spannung ist jetzt mit dem nötigen Strom belastbar und über einen Analogeingang (ANn) durch die Software im 80535 überprüfbar. Auf diese Weise werden zwei verschiedene Programmierspannungen (Up1 und Up2) und die Versorgungsspannung für die zu programmierenden ICs (Uccp) erzeugt. Die Unterschiede der drei Schaltungsteile liegen lediglich bei den je zwei Operationsverstärkern. Zu Uccp kann man über das Bit SUcg wahlweise 5 Volt addieren. Zu Up1 und Up2 werden konstant 5 Volt addiert.

Alle drei Spannungen können also über einen BD 135 von einem Prozessorbit geschaltet werden. Geschieht dies, so ändert sich die Spannung am Programmiersockel mit ca. 1,5 V/µs. Für positive Flankensteilheiten von ca. 0,4 V/µs sind bei den Programmierspannungen (Up1, Up2) die abschaltbaren Kapazitäten C6 und C10 vorhanden. Sie werden nur bei einigen PROMs benötigt. Bei höheren Flankensteilheiten werden die Schaltvorgänge mit den Transistoren in den noch zu beschreibenden Programmierzellen realisiert.

Die Programmierbits

Um die Spannungen am Programmiersockel zu steuern und um die Signale an den Pins zu lesen, werden viele Port-Bits am 80535 benötigt (*Tabelle 4*). Die Prozessorbits (Px.y) reichen dazu nicht aus. Die zusätzliche Steuerbits (XPx.y) sind mit den Zwischenspeichern IC5 bis IC11 realisiert worden. Weiter gibt es die Leseports IC12 bis IC14 (RPx.y). Das IC4 wählt anhand der Adreßleitungen A3 und A4 ein Port aus und veranlaßt mit einem seiner Ausgänge den Datentransport von oder zu den entsprechenden Bausteinen. Die Bezeichnung setzt sich aus der physikalischen Adresse x und dem Bit y zusammen.

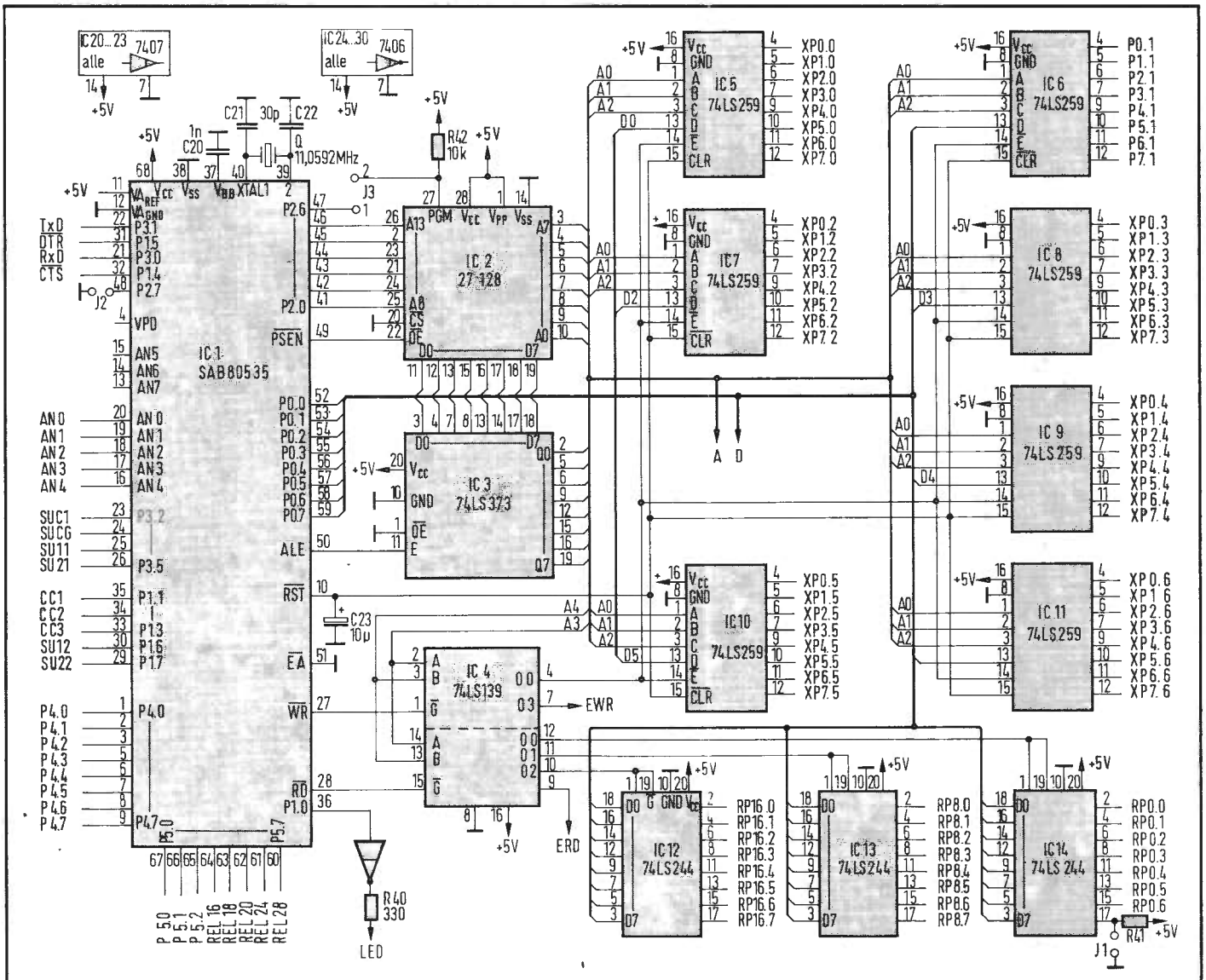


Bild 7. Der Ein-Chip-Rechner 80535 steuert, mißt und regelt alles in mc-Allprog

Tabelle 4: Die Portbits der CPU und die Signalnamen bei den Programmierzellen

Pin n	Bit a	Bit b	Bit c	Bit d	Pull-up-Widerstand
1	XP7.5	XP7.6			R101
3	P4.6	P4.7			R103
5	P4.4	P4.5		RP16.7	R105
6	P4.2	P4.3		RP16.6	R106
7	P4.0	P4.1		RP16.5	R107
8	XP6.5	XP6.6		RP16.4	R108
9	XP5.5	XP5.6		RP16.3	R109
10	XP4.5	XP4.6		RP16.2	R110
11	XP7.3	XP7.4		RP16.1	R111
12	XP6.3	XP6.4		RP16.0	R112
13	XP5.3	XP5.4		RP8.0	R113
14	XP4.3	XP4.4		RP8.1	R114
15	XP3.3	XP3.4	XP3.5	RP8.2	R115
17	P5.0	P5.1	P5.2	RP8.3	R117
18	XP0.0	XP0.1	XP0.2	RP8.4	R218
19	XP1.0	XP1.1	XP1.2	RP8.5	R219
20	XP2.0	XP2.1	XP2.2	RP8.6	R220
21	XP3.0	XP3.1	XP3.2	RP8.7	R221
22	XP4.0	XP4.1	XP4.2	RP0.0	R222
23	XP5.0	XP5.1	XP5.2	RP0.1	R223
24	XP6.0	XP6.1	XP6.2	RP0.2	R224
25	XP7.0	XP7.1	XP7.2	RP0.3	R225
26	XP0.3	XP0.4	XP0.5	RP0.4	R226
27	XP1.3	XP1.4	XP1.5	RP0.5	R227
28	XP2.3		XP2.5	RP0.6	R128

Tabelle 5: Der Zusammenhang zwischen Pin-Nummern der Fassung und Pin-Nummern der Gehäuse

Pin n	28	24	20	18	16
1					
2					
3	1				
4	2				
5	3	1			
6	4	2			
7	5	3	1		
8	6	4	2	1	
9	7	5	3	2	1
10	8	6	4	3	2
11	9	7	5	4	3
12	10	8	6	5	4
13	11	9	7	6	5
14	12	10	8	7	6
15	13	11	9	8	7
16 GND	14	12	10	9	8
17	15	13	11	10	9
18	16	14	12	11	10
19	17	15	13	12	11
20	18	16	14	13	12
21	19	17	15	14	13
22	20	18	16	15	14
23	21	19	17	16	15
24	22	20	18	17	16
25	23	21	19	18	
26	24	22	20		
27	25	23			
28	26	24			
29	27				
30	28				
31					
32					

Am Prozessorbus befinden sich noch weitere, fast selbstverständliche Bausteine: Der Programmspeicher (EPROM IC2), ein Latch für die Adressen A0 bis A7 (IC3) und der Stecker ST2. Dieser Stecker dient momentan nur zu Servicezwecken. Über ihn ist aber auch eine Erweiterung des Gerätes denkbar. So könnte man ein Display, eine Tastatur und Speicher anschließen, um ohne den Anschluß eines externen Rechners arbeiten zu können. Doch das ist noch Zukunftsmusik.

Die Programmierzellen

Zuerst eine Festlegung: Wenn im Text von einem Pin n des Programmiersockels die Rede ist, so bezieht sich diese Nummer immer auf dessen Numerierung 1 bis 32. Die tatsächliche Pinnummer für 16-, 18-, 20-, 24- oder 28-Pin-

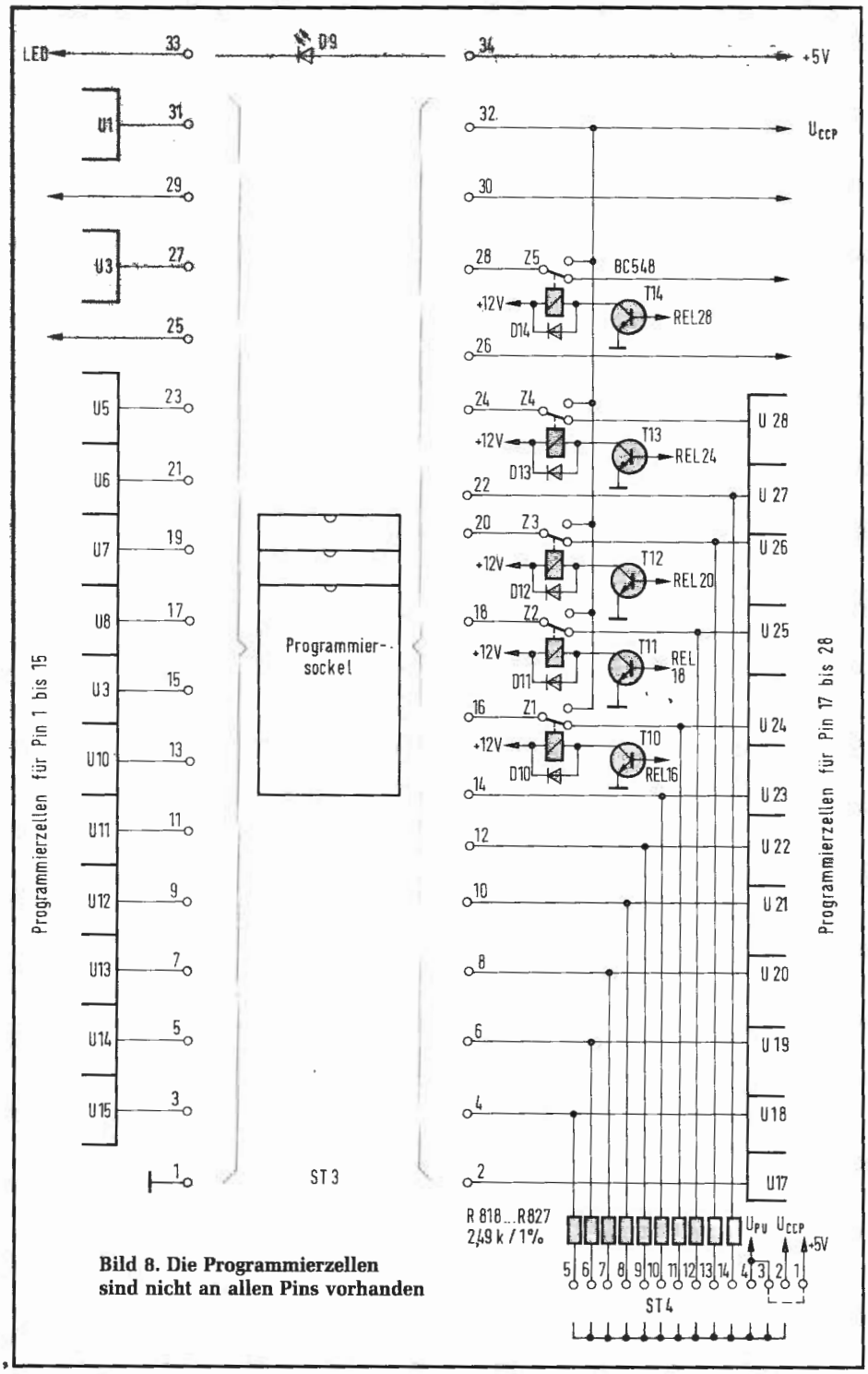


Bild 8. Die Programmierzellen sind nicht an allen Pins vorhanden

Bausteine ist natürlich eine andere. Einen Überblick verschafft Tabelle 5. Bei der Arbeit an den zu programmierenden ICs müssen die einzelnen Pins der Programmierfassung gezielt auf logisch 0, logisch 1, eine der beiden Programmierspannungen (Up1 oder Up2) gelegt werden – oder ihr Zustand soll eingelesen werden. Dafür sind die sogenannten Programmierzellen konstruiert. Da diese Beschaltung für die meisten Pins übereinstimmt, wurde zugunsten der Übersichtlichkeit auf einen Gesamtschaltplan verzichtet. An welchem Pin eine solche Programmierzelle angeschlossen ist, zeigen *Bild 8* und Tab. 4. Die in *Bild 9* gezeigte Programmierzelle ist nicht immer komplett vorhanden. Fehlt in der Spalte Bit b (Tabelle 4) ein Eintrag, so sind zu diesem Pin n die Bauteile R3n, R4n, T1n und der dazugehörende Treiber nicht vorhanden. An

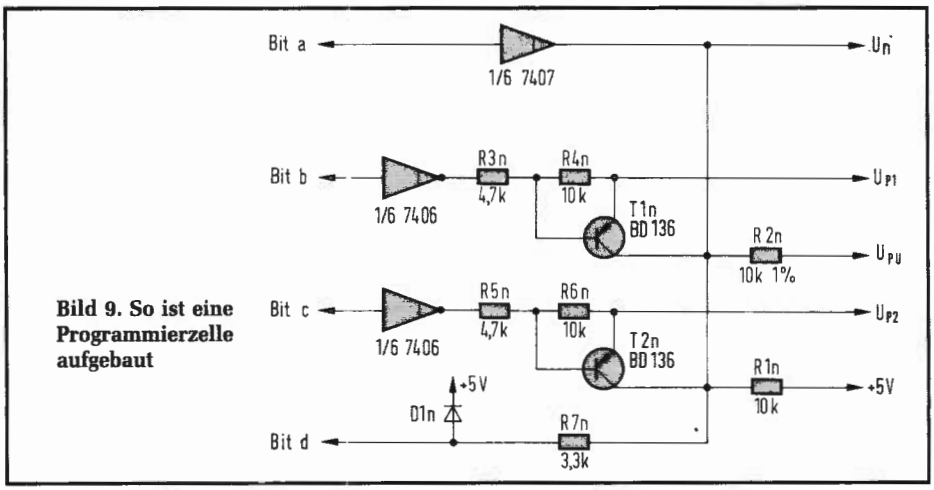


Bild 9. So ist eine Programmierzelle aufgebaut

diesem Pin kann dann die Programmierspannung Up1 nicht geschaltet werden. Entsprechendes gilt für die Spannung Up2 anhand von Spalte Bit d in Bezug auf die Bauteile R5n, R6n, T2n und einen Treiber. Die Leseports (RPx.y) wer-

Tabelle 6: Die Stückliste

Anzahl	Elemente	
	Integrierte Schaltungen	
1	IC1	SAB 80535
1	IC2	27128
1	IC3	74LS373
1	IC4	74LS139
7	IC5–11	74LS259
3	IC12–14	74LS244
1	IC15	78S05
1	IC16	75188
1	IC17	75189
2	IC18, 19	RC4136
4	IC20–23	7407
7	IC24–30	7406
	Quarz	
1	Q	11,0592 MHz
	Klemme	
1	K1	2pol. Klemme
	Platinen und Zubehör	
1	Platine mc-AllProg V1.0	
1	kleine Platine für Programmiersockel und LED	
1	Programmiersockel	
1	PLCC-68-Sockel	
1	34pol. Platinenverbinder	
1	34pol. Stecker (für ST3)	
1	14pol. Stecker (für ST4)	
1	Flachbandkabel	
3	Glimmerscheiben (für T3, 6, 9)	
1	Kühlblech (50 mm breit)	
	Stecker	
1	ST1	2× 5pol., Pfosten
2	ST2, 4	2× 7pol., Pfosten
1	ST3	2× 17pol., Pfosten
	Jumper	
3	J1–3	2pol., Pfosten
	Relais	
5	Z1–5	1× um, 12 V

Anzahl	Elemente	
	Transistoren	
3	T2, 5, 8	BC177
5	T10–14	BC548
6	T1, 3, 4, 6, 7, 9	BD135-16
37	T101, 103, 105–115, 117–127, 215, 217–228	BD136-16
	Dioden	
1	D2	1N4001
30	D7, 8, 10–14, 105–115, 117–128	1N4148
1	D1	1N5401
4	D3–6	AA119
1	D9	LED rot 1,8 mm
	Kondensatoren	
2	C21, 22	30 pF, keramisch, RM 5
3	C6, 10, 20	1 nF, keramisch, RM 5
8	C3, 7, 11, 16, 24–27	100 nF, RM 5
6	C13, 14, 28–31	1 µF Tantal, 16 V, stehend
7	C1, 2, 4, 5, 8, 9, 12	4,7 µF Elko, 16 V, stehend
1	C23	10 µF Elko, 16 V, stehend
1	C17	100 µF Elko, 16 V, stehend
1	C15	100 µF Elko, 40 V, stehend
	Widerstandsnetzwerke	enthält Widerstand:
3	RW1–3 DIL 8 × 3,3 kΩ	R705–712; R713–715, 717–721; R722–728, 41
4	RW4–7 DIL 8 × 4,7 kΩ	R301, 303, 305–310; R311–315, 317, 326, 327; R318–325; R518–525
6	RW8–13 SIL 8 × 10 kΩ	R101, 103, 105–109, 42; R110–115, 117, 128; R401, 403, 405–410; R411–415, 417, 426, 427; R418–425; R618–425
	Widerstände	
1	R1	(siehe Text)
8	R7, 11, 13, 18, 22, 28, 32, 40	330 Ω
1	R3	1,00 kΩ 1%
3	R15, 25, 35	1,82 kΩ 1%
10	R818–827	2,49 kΩ 1%
5	R515, 517, 526–528	4,7 kΩ
3	R14, 24, 34	7,50 kΩ 1%
1	R2	9,09 kΩ 1%
10	R218–227	10 kΩ 1%
13	R4, 8, 9, 19, 20, 29, 30, 36, 615, 617, 626–628	10 kΩ
6	R10, 12, 21, 23, 31, 33	56 kΩ
6	R5, 6, 16, 17, 26, 27	100 kΩ
1	R37	10 kΩ NTC

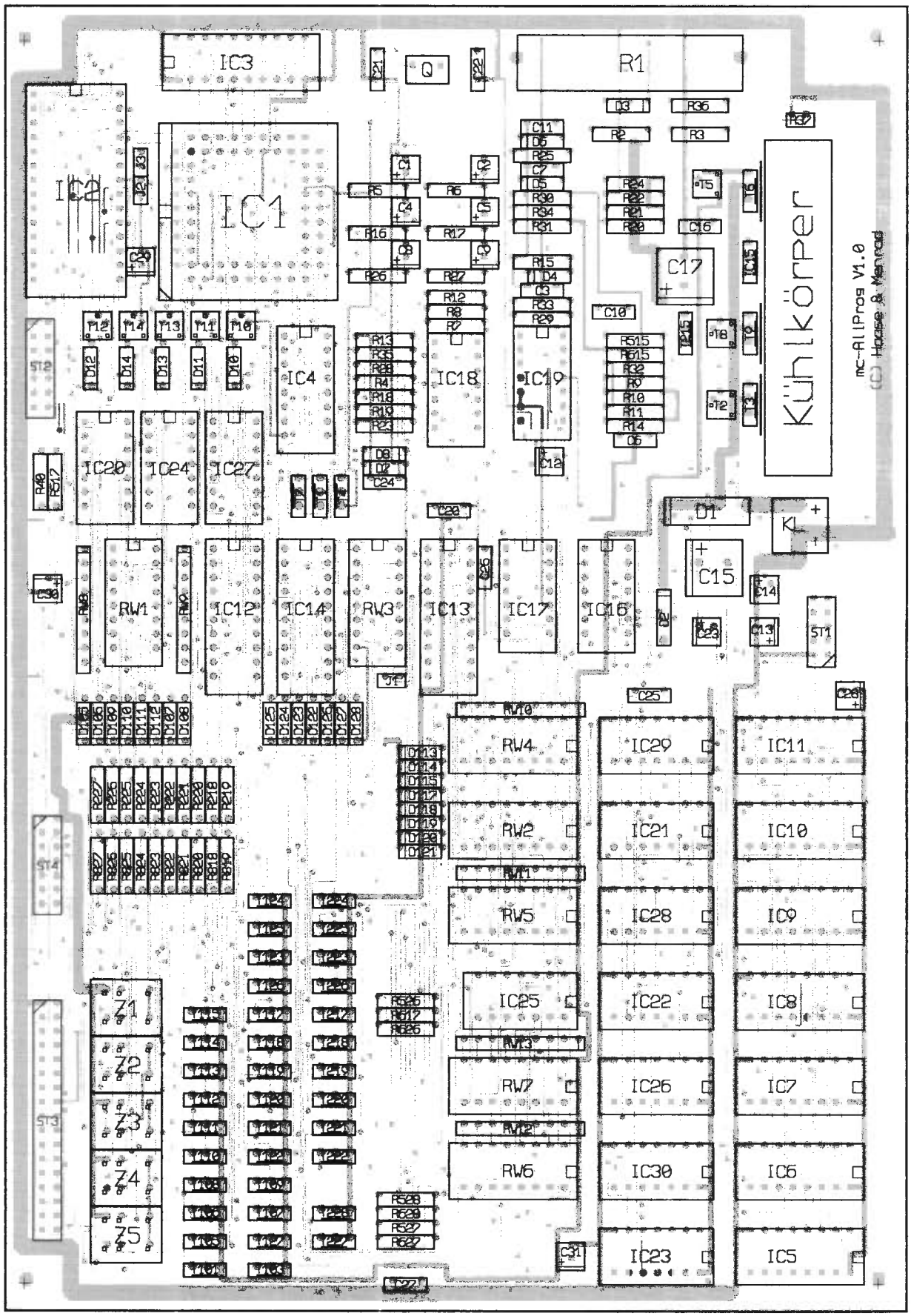


Bild 10. Der Bestückungsplan zeigt den gedrängten Aufbau

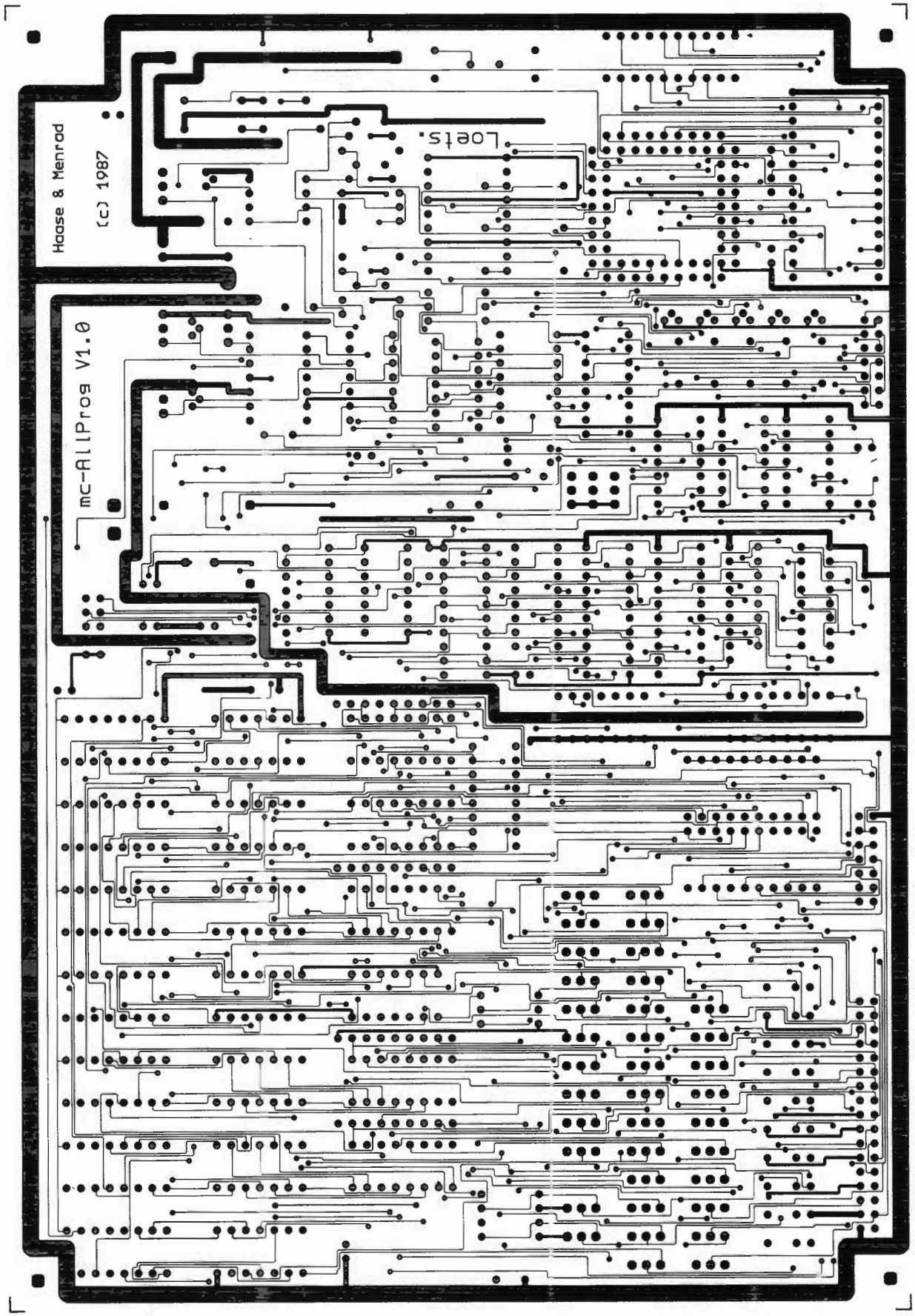


Bild 11. Die Lötseite ...

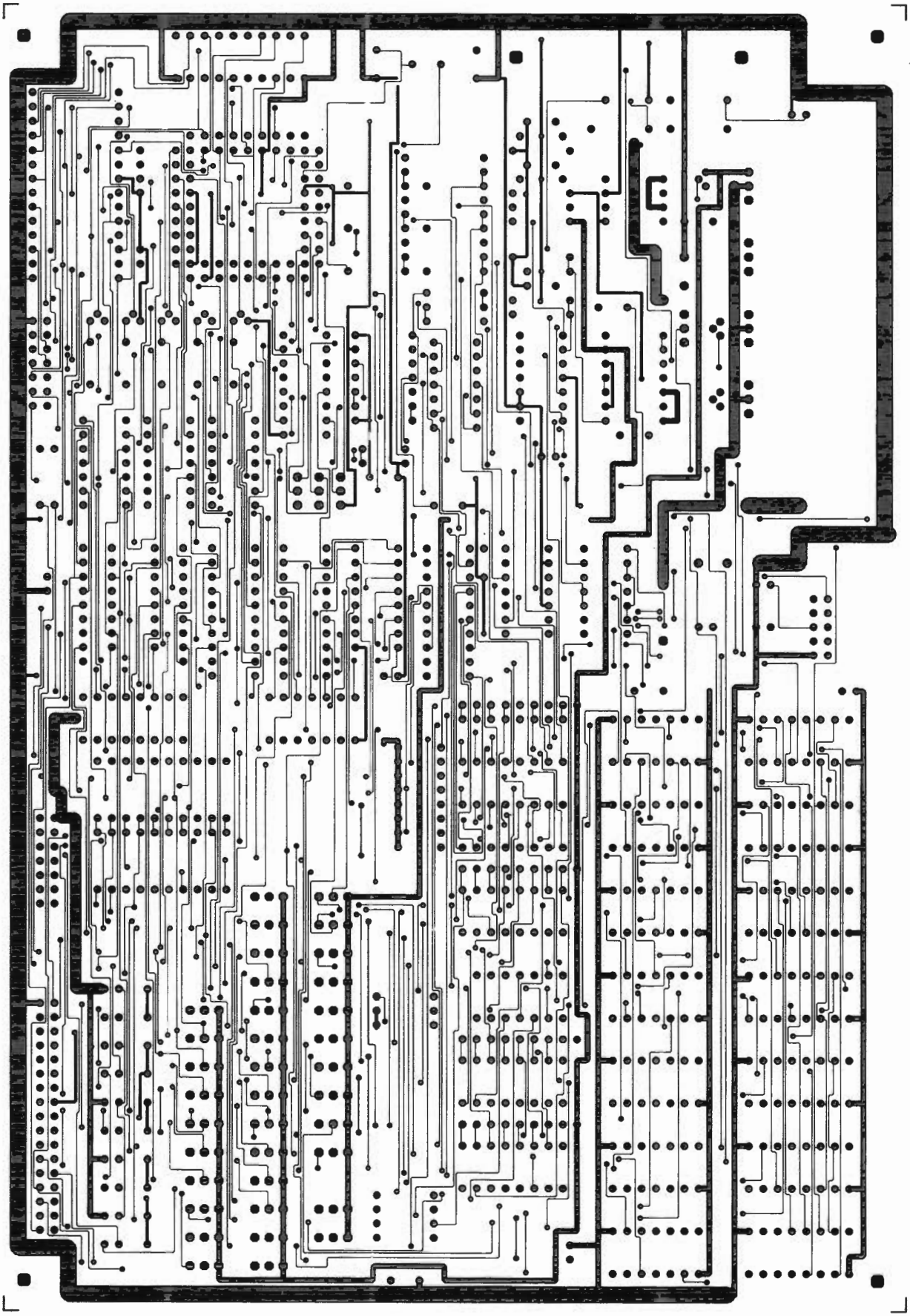


Bild 12. Die Bestückungsseite der Platine

den durch die Diode D1n und den Widerstand R7n vor Spannungen über 5 Volt geschützt. Fehlt eines dieser Bits in der Spalte Bit d, so fehlen natürlich auch diese beiden Bauteile in der Schaltung. Alle Pins sind mit einem Pull-up Widerstand von 10 kOhm an +5 Volt gelegt. Dies ist z.T. direkt mit R1n oder über den Stecker ST4 mit R2n realisiert. Es existiert je Zelle nur einer dieser beiden Widerstände R1n oder R2n. Am Pfostenfeld ST4 verbindet ein Jumper die Pfosten 1 und 3, damit auch an den Widerständen R2n die gewünschten +5 Volt liegen. Dieser Jumper wird nur bei einer Sorte ICs durch den in Bild 7 gezeigten Stecker ersetzt: bei PALs der Firma Advanced Micro Devices (AMD). In diesem Fall entstehen durch die Parallelschaltung von R2n und R8n an den Pins 18 bis 27 der Programmierfassung, Pull-Up-Widerstände mit je 2 kOhm zu Uccp.

Wird die Platine in ein Gehäuse eingebaut, so ist es ratsam, an ST4 einen Schalter anzuschließen. Dieser kann dann zwischen der „Jumper" und der „Stecker"-Konfiguration umschalten. Um die Montage der Platine zu vereinfachen und die Bauteilmenge zu reduzieren wurden viele Widerstände der Programmierzellen zu Netzwerken zusammengefaßt. Dies wird im Bestückungsplan (*Bild 10*) und der Stückliste (*Tabelle 6*) verdeutlicht.

Der Programmiersockel

Die Aufnahme der zu bearbeitenden Bausteine wird mit einem der für EPROMs oft verwendeten Null-Kraft-Sockel realisiert. Der vorgeschlagene Sockel hat sehr breite Kontaktfedern, so daß auch schmale 16-Pin-PROMs festgeklemmt werden können. Er befindet sich zusammen mit einer Leuchtdiode (die die Betriebszustände signalisiert) auf einer kleinen Platine, die über ein Flachbandkabel mit dem Pfostenstecker ST3 verbunden ist. Die genannte Fassung ist leider nur 28polig verfügbar. Ein Argument gegen diesen Sockel: Werden viele ICs nur einer Breite bearbeitet, so zeigen sich schnell Ermüdungserscheinungen der Kontaktfedern. Abhilfe kann das Parallelschalten eines schmalen 24poligen mit einem breiten 28 poligen Sockel schaffen. An dieser Stelle muß allerdings vor beliebig vielen Fassungen oder zu langen Zuleitungen zu diesen Programmiersockeln gewarnt werden. Sonst entstehen zu große Leitungskapazitäten und Reflexionen zwischen den Bausteinen und der Programmierkarte, die zu Zerstörungen führen können.

Auch die 32poligen EPROMs benötigen eine gesonderte Einheit aus „Fassung, LED, Platine, Kabel". Auf gleiche Weise sind auch leicht Programmiermöglichkeiten für 40-Pin-Kontroller mit eingebautem EPROM (z.B. 8748 oder 8751) denkbar.
Die Bausteine werden immer so in dem Sockel befestigt, daß ihr Massepin, unabhängig von ihrer Pinanzahl, in derselben Ecke liegt. Deshalb gibt es die Relais Z1 bis Z5. Sie schalten für ICs unterschiedlicher Größen die Versorgungsspannung Uccp. Die Steuerung der Relais geschieht über die Transistoren T10 bis T14 durch die Prozessorbits Rel16 bis Rel28.

Einschalten und Betrieb

Nach dem Einschalten blinkt die Leuchtdiode neben dem Programmiersockel langsam im Gleichtakt. Dies signalisiert einen Prüf- und Kalibriervorgang. Dabei werden die Bauteiltoleranzen der DA-Wandler ausgemessen. Auf diese Weise wird manuelles Einstellen von Spannungen, wie es bei den meisten Programmiergeräten nötig ist, überflüssig. Einige Sekunden nach dem Einschalten verändert sich das Blinken der LED. Es sind drei Zustände möglich:

- 3er Blinken: die externe Programmierspannung ist zu klein oder Bauteile sind defekt.
- 2er Blinken: Fehler im Handshaking der seriellen Verbindung (z.B. fehlende Betriebsbereitschaft an CTS).
- gleichmäßiges Leuchten: Die Hardware von mc-AllProg ist betriebsbereit.

Im Betriebszustand des Gerätes hat diese LED stets folgende Bedeutung:

- gleichmäßiges Leuchten: Der Zustand des Programmiersockels darf nicht verändert werden (Keine IC einlegen oder herausnehmen)!
- dunkel: IC in Programmiersockel einlegen und festklemmen!
- Blitzen: IC darf oder muß aus dem Programmiersockel genommen werden!

Nun noch ein paar Worte zur Entstehung dieses Gerätes: Angesichts der hohen Kosten, die durch Fehlprogrammierung von nicht löschbaren Bausteinen entstehen können, haben wir uns viele Monate Zeit genommen, um einerseits die Programmierspezifikationen der IC-Hersteller genau einzuhalten und außerdem den Benutzer vor Programmierfehlern zu bewahren. Leider kostete der Umgang mit dem Prozessor der Firma Siemens sehr viel Zeit. Das zu ihm verfügbare „User's Manual 7.85" ist der Realität der uns gelieferten Chips in einigen Punkten weit voraus. Bausatz und Software dazu sind bei den Autoren erhältlich.

Fortsetzung folgt, mit Programmierprinzipien, Bausteinbeschreibungen und der Ansteuerung von Allprog.

Know-how-Computer

Computer für Anfänger heißt ein Paket aus Anleitung, papiernem Computermodell (Know-how-Computer) und Papier-CPU. Diese drei Objekte aus Papier bekommt jeder umsonst (Porto von 2,50 DM muß allerdings der Bestellung beigefügt sein), der es unter dem Stichwort „Know-how-Computer" beim Franzis-Verlag bestellt. Der Inhalt ist bewährt. Er hat schon sehr vielen Lesern zu genauen Kenntnissen darüber verholfen, wie Computer funktionieren. Man kann nämlich mit der Anleitung und Papierformularen zunächst anhand des Know-how-Computers das algorithmische Arbeiten von Computern kennenlernen und dann anhand der Papier-CPU noch exakt durchspielen, wie die CPU des Know-how-Computers arbeitet. Übrigens sind diese Objekte in Zusammenarbeit mit dem WDR-Wissenschaftsredakteur Wolfgang Back entwickelt worden. Lehrer können unter der Tel.-Nr.

089/5117-337 (es meldet sich Frau Reise) nach Klassensätzen für den Unterricht fragen.

H. Nikolaus Schaller

Z80-Karte für den mc-68000-Computer

Teil 2: Inbetriebnahme und Software

Nachdem im ersten Teil Schaltung und Aufbau der Z80-Karte beschrieben wurden, geht es nun in die Paxis: Neben Tips zur Inbetriebnahme und Fehlerbeseitigung findet sich in diesem Teil eine ausführliche Beschreibung der prinzipiellen Arbeitsweise der CP/M-80-Emulator-Software.

Zum Lieferumfang der Z80-Karte gehört eine CP/M-68k-Diskette mit Programmen für Initialisierung, Test und Betrieb der Karte. Ein vollständiger Abdruck all dieser Dateien würde den Rahmen dieses Beitrags bei weitem sprengen, wir setzen daher im folgenden das Vorhandensein dieser Diskette voraus (die auch mit der Leerplatine vom Hersteller geliefert wird) und beschränken uns auf die Wiedergabe der wesentlichen und für das Verständnis der Funktion wichtigen Programmteile.

Schritt für Schritt

Nach der Bestückung der Z80-Karte (ohne ICs, *Bild 6...8*) sollte eine genaue Sichtinspektion folgen, um Lötzinnbrücken und nicht angeschlossene IC-Pins auszuschließen. Anschließend wird mit einem Ohmmeter zwischen den Anschlüssen der Betriebsspannung (z.B. Pin 7 und Pin 14 eines TTL-ICs) geprüft, ob kein Kurzschluß vorliegt. Der Widerstand sollte unmeßbar groß sein, da es keine Spannungsteiler gibt. Ein kurzzeitig niedriger Widerstand, der dann ansteigt, kommt von den Kondensatoren, die sich über das Meßgerät aufladen.
Jetzt kann man die Platine zum ersten Mal in einen Steckplatz des mc-68000-Computers einsetzen, wobei die Steckplätze 0 oder 1 nicht verwendet werden sollten, da sie eine Sonderbedeutung im Betriebssystem haben. Beim Einschalten des Rechners dürfen sich keine Unterschiede zu früher zeigen. Meldet sich der mc-68000-Computer nicht mehr, dann ist ein Kurzschluß der Busleitungen zu vermuten – ein Fehler, der unbedingt behoben werden muß, bevor man weitermacht.
Anschließend sollte ein großes, rechenintensives Programm gestartet werden, zum Beispiel der Assembler des CP/M-68K. Stürzt der mc-68000-Computer dabei ab, deutet das auf Probleme mit dem 68000-Bus hin. Eine Terminatorkarte – falls noch nicht vorhanden – schafft hier Abhilfe.
Nun kann man die Z80-Karte mit ICs bestücken, einstecken und den Rechner wieder einschalten (Karten sollten nur bei abgeschalteter Betriebsspannung ein- und ausgesteckt werden!). Zeigt der mc-68000-Computer jetzt Fehlfunktionen, kommen mehrere Ursachen in Frage: Zunächst kann das Netzteil überlastet sein, was sich mit einem Spannungsmeßgerät leicht prüfen läßt – auf der Z80-Platine direkt an den ICs müssen 4,75...5,25 V liegen. Auch kann die Busbelastung durch die Treiber zu hoch geworden sein. Hier hilft ebenfalls die Terminatorkarte. Tritt ein Busfehler auf, dann ist entweder die Steuerung nicht in Ordnung oder der Rechner läuft noch mit einer alten Betriebssystemversion (1.3 oder früher), die ersetzt werden muß. Meldet der Rechner jedoch eine „Exception $0F", so bedeutet das, daß der Z80-Prozessor abgestürzt ist und dabei einen Interrupt beim 68000 ausgelöst hat. Das kann an der EPROM-Zugriffszeit, am RAM oder am Z80 selbst liegen. Schließlich ist ein Defekt auf der Z80-Karte selbst möglich. Sind Lötfehler ausgeschlossen, dann bringt eine Eingrenzung des Fehlers durch Austausch möglicherweise defekter Treiberbausteine im 68000-Interface meist Erfolg.

Initialisierung

Verlief die Inbetriebnahme bisher einwandfrei, kann man die Z80-Karte unter CP/M-68K mit dem Programm Z80INIT.68K an ihrem Steckplatz initialisieren.
Für die folgenden Tests benötigt man die Basisadresse der Karte. Um sie herauszufinden, startet man das Programm DDT.68K ohne Parameter und erhält mit „dlfff800" die Speicherbelegungstabelle. Der Eintrag für die Z80-Karte beginnt mit dem Wert $06020423. Im folgenden Langwort steht die Basisadresse der Karte. Solange sich die Belegung der Steckplätze nicht ändert, behält diese Adresse ihre Gültigkeit. Die folgenden Adreßangaben sind zu dieser Adresse zu addieren, da sie sich auf den Z80-Adreßraum beziehen.
Mit „dw" ab Adresse 0 kann man nun prüfen, ob die Z80-Karte bei Wortzugriffen einen Busfehler erzeugt. Wenn nicht, funktioniert der Decoder nicht (IC 19, IC 26 oder IC 30).
Beim Auslesen einiger Bytes ab der Basisadresse mit „d" kann einer der folgenden Fehler auftreten:

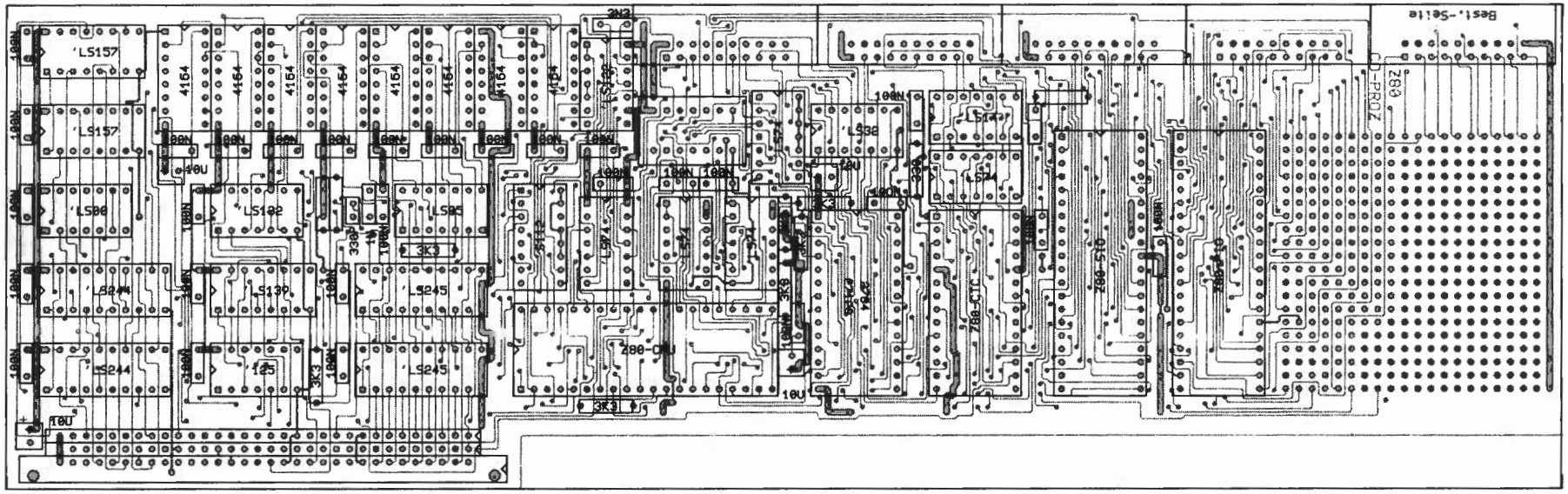
- Meldet sich der 68000 nicht mehr, dann wird kein DTACK erzeugt. Das kann daran liegen, daß der Z80 keinen Takt bekommt (IC 11, IC 12, IC 15), daß die CPU keine Busanforderung erhält (IC 14, IC 21) oder die Speichersteuerung nicht funktioniert (IC 26, IC 18, IC 22, IC 19).

- Haben die ersten Bytes immer den Wert $FF, so funktioniert die DTACK-Erzeugung, der Z80-Speicher oder die Datentreiber aber nicht. Als weitere Ursache kommen auch die EPROM-Dekodierung oder das EPROM selbst in Frage.

- Zeigen die ersten Bytes ein regelmäßiges Muster (z. B. 00 39 00 39), so läuft der Z80 nicht korrekt. Die Werte zeigen, daß das EPROM funktioniert, und abgeschaltet wird. Danach stürzt der Z80 aber ab, und überschreibt den gesamten Speicher.

- Steht dort „06 00 10 FE" usw., dann läßt sich das EPROM vom 68000 aus lesen, der Z80 hat es aber nicht abgeschaltet. Als Hauptursache dafür kommt der dynamische Speicher in Frage, eine weitere Möglichkeit wird im nächsten Absatz beschrieben.

Bitte lesen Sie weiter Seite 54


LS157
LS157
LS86
LS244
LS244
LS32
LS138
LS245
LS245
LS74
LS74
LS74

Bei richtiger Initialisierung des Speichers lauten die ersten Bytes „C3 03 FE 00 00 C3 06 FD". An der Adresse $0FD00 müssen nun folgende Bytes stehen:
00 00 00 00 00 00 C3 11 FD 00 00
Das bedeutet, daß der Z80 den EPROM-Inhalt richtig ins RAM kopiert hat. Stehen hier andere Werte, so muß die Speichersteuerschaltung überprüft werden. Dazu muß man den Rechner mit Reset neu starten und die Zeitbedingungen des RAS-Signals (Tabelle 5 und Diagramm 1 im ersten Teil) mit einem Oszilloskop kontrollieren. Dabei sollte man auch die anderen dort angegebenen Parameter messen. Weichen die Werte zu sehr ab, müssen die RC-Glieder geändert werden. Stimmen die übertragenen Bytes, das EPROM wird jedoch nicht abgeschaltet, so liegt der Fehler bei IC 24 oder IC 26.
Jetzt kann man mit dem s-Befehl versuchen, ob sich der Speicher beschreiben läßt. Bei Problemen kann auch das Programm ZRAMTEST eingesetzt werden. Es überprüft die wesentlichen Funktionen des Z80 und seines Speichers. Bei Fehlern wird gemeldet, in welchem Bereich der Schaltung sie auftreten. Melden sich CP/M-80-Programme überhaupt nicht, obwohl ZRAMTEST keine Fehler meldete, kommt nur noch ein Fehler in der Interrupterzeugung (IC 19, IC 21 oder BE) in Frage.

Test der Peripherie

Mit einem Debugger lassen sich die Peripheriebausteine testen. Für die folgenden Beispiele wird die Verwendung des DDTZ-Debuggers angenommen; die Peripheriebausteine müssen natürlich eingesetzt sein. Treten gleich nach dem Starten des Debuggers (A>z80 ddtz) Fehler auf, dann sind die Peripheriebausteine die Ursache. In den SIO-Statusregistern (i2, i3) sollte 44, 4C, 6C, 64 oder ein ähnlicher Wert stehen. Nachdem man einen beliebigen Wert (z. B. $55) zweimal in beide Sendedatenregister geschrieben hat (o55 0, o55 0, o55 1, o55 1), sollte das zweite Bit im Statusregister (Tx Buffer empty) gelöscht sein (40, 48, 68, 60).
Den PIO-Baustein kann man testen, indem beide Ports als Ausgang programmiert werden, ein Wert (z.B. $55,$aa) geschrieben und zurückgelesen wird (of 42, of 43, o55 40,oaa 41, i40, i41). Stimmt das Zurückgelesene, dann funktioniert die PIO. Man kann zusätzlich die Pegel am Peripheriestecker messen.
Am schönsten ist der CTC zu testen: Timer 0 wird dazu auf freilaufend, Teilfaktor 256 und Zeitkonstante 200 eingestellt. Dazu wird der CTC-Kanal 0 (Adresse 80H) mit dem Steuerbyte 025H und der Zeitkonstanten 0C8H (o25 80, oc8 80) programmiert. Wird jetzt mehrmals das Zählerregister gelesen (i80, bzw. i allein), sollte jedesmal ein anderer Wert erscheinen, der jedoch nie größer als 0C8H wird. Dieser Test muß auch bei den anderen Kanälen erfolgreich sein.
Um die Z80-Interrupts zu testen, gibt es das Hilfsprogramm IRQ.COM. Es programmiert den CTC wie oben, schaltet aber zusätzlich den Interrupt ein. Jedesmal, wenn der Timer abläuft (bei 6 MHz Takt etwa alle 8,5ms), wird eine Speicherzelle dekrementiert. Wenn sie Null wird, gibt das Programm über einen BIOS-Aufruf ein „x" aus. Das geschieht bei 6 MHz Taktfrequenz alle 2,2 Sekunden. Treten während dieser Tests oder später sporadische Fehler auf, etwa ein Steckenbleiben des Z80 oder fehlerhafte BDOS-Aufrufe, deutet dies auf einen unregelmäßigen Speicherverlust hin. Es kann daran liegen, daß der Z80 über WAIT zu lange angehalten wird, es ist auch möglich, daß die Zeitbeziehungen der Speichersteuerung an der Grenze liegen (Diagramm 1 und Tabelle 5 im ersten Teil). Zudem kommen auch Störungen auf der Betriebsspannung oder auf dem Bus als Ursache in Frage. Wenn ZRAMTEST keinen Fehler bei der Z80-Initialisierung meldet, bei manchen oder allen Durchläufen jedoch Speicherfehler findet, die jedesmal an einer anderen Adresse auftreten und bei denen z.B. $00, $FF oder Regelmäßiges wie $80, $C0 oder $E0 gelesen wird, unabhängig davon, welcher Wert in der Zelle stehen sollte, dann liegt das an Störungen, die das Businterface durcheinanderbringen. Dagegen hilft ein Kondensator mit 100...150 pF zwischen Pin 11 und Pin 14 von IC 21 (auf der Platinenunterseite anlöten). Dadurch werden Schwingungen des SSEL-Signals unterdrückt.

Programmierung des Z80

Nach jedem Reset führt der Z80 ein Initialisierungsprogramm aus dem EPROM aus. Dieses kopiert einen Teil des EPROM-Inhalts an das obere Ende (ab 0FD00H) des RAMs und springt dorthin. Nun wird zuerst das EPROM abgeschaltet, dann der Stackpointer initialisiert, anschließend das Startflag gelöscht und schließlich auf Aufträge gewartet. Der 68000 kann den Z80 dazu veranlassen, ein Programm ab Adresse 100H als Unterprogramm auszuführen, indem er das Startflag auf einen von Null verschiedenen Wert setzt. Ist das Unterprogramm beendet, springt der Z80 nach 0FE03H. Das bedeutet soviel wie einen BIOS-Warmboot-Aufruf. In Diagramm 3 (*Bild 9*) ist dieser Ablauf graphisch dargestellt.

Z80 unterbricht 68000

Bei einem BIOS- oder BDOS-Aufruf des Z80 werden die Parameter, die in Z80-Registern stehen, im Speicher abgelegt, damit sie der 68000 erreichen kann. Dann wird das Flag unter Adresse 0FFF1H bei BDOS-Aufrufen auf Null gesetzt, bei BIOS-Aufrufen auf einen von Null verschiedenen Wert. Als nächstes wird der IRQ2 eingeschaltet. Ob der 68000 reagiert hat, wird daran erkannt, daß in Zelle 0FFF8H (Interruptflag) nicht mehr Null steht. Wenn diese Situation eintritt, wird der Interrupt wieder abgeschaltet. Falls innerhalb einer halben Sekunde keine Bestätigung kommt, stoppt der Z80 völlig.
Reagiert der 68000 jedoch, so wartet der Z80, bis der 68000 seine Arbeit beendet, was er am gelöschten Interruptflag feststellt. Dann lädt der Z80 die Ergebnisse in seine Register und setzt seine Arbeit im aufrufenden Hauptprogramm fort. Erkennt der 68000 das Ende eines Z80-Programms (BDOS-Funktion 0, BIOS-Warmboot) oder tritt ein Fehler auf, so bleibt der Z80 in der Warteschleife, die erst unterbrochen wird, wenn der 68000 ein neues Z80-Programm laden will und dazu einen Reset auslöst. Diese Abläufe sind in Diagramm 4 (Bild 9) dargestellt.
Falls eigene Applikation eine andere Art der Parameterübergabe benötigt, kann man entweder andere BIOS-Aufrufe definieren und festgelegte Speicherbereiche zum Datenaustausch verwenden – oder sowohl das Programm im EPROM als auch das 68000-Steuerprogramm ändern.

Die Z80-Peripherie

Die Adressen, unter denen die Peripheriebausteine angesprochen werden können, stehen in Tabelle 2 im ersten Teil dieses Beitrages. Bei 0C0H bis 0C3H können sowohl EPROM als auch der 68000-Interrupt ein- und ausgeschaltet werden. Daß beide gekoppelt sind, stört nicht, da ein Programm, das einen Interrupt auslöst, wohl „weiß", ob es im EPROM oder im RAM steht. Wenn das EPROM abgeschaltet wird, ist normalerweise auch kein Interrupt aktiv. Das EPROM kann jederzeit wieder aktiviert werden, nur muß berücksichtigt werden, daß damit die untere RAM-Hälfte nicht mehr erreichbar ist: das Umschaltpro-

gramm muß in der oberen Hälfte stehen. Beim PIO-Baustein ist zu beachten, daß er kein Reset-Signal erhält. Es kann daher passieren, daß er sich in einem undefinierten Zustand befindet, wenn ein Programm gestartet wird. Zur Initialisierung schreibt man am besten ein Nullbyte in das Steuerregister. Das ist bei Steuersequenzen, die mehr als ein Byte benötigen, das zweite Byte, sonst wird es als Interruptvektor verwendet, den man danach gegebenenfalls auf den benötigten Wert setzen können.

Die automatische WAIT-Funktion beim SIO-Kanal A kann durch Register WR1 eingeschaltet werden. Sie dient der schnellen Datenübertragung durch Block-I/O-Befehle (z.B. OTIR) sowie der Initialisierung. Im Übrigen sei auf die im Buchhandel erhältlichen Z80-Programmieranleitungen verwiesen.

CP/M-80-Emulation

Auf dem mc-68000-Computer läuft CP/M-68K, das bis auf Kleinigkeiten CP/M-80-kompatibel ist. Werden nun alle Systemaufrufe vom Z80 zum 68000 geleitet und dabei die Unterschiede kompensiert, so entsteht für den Z80 eine CP/M-80-Umgebung, ohne dieses Betriebssystem selbst in den Speicher zu laden. Diese Anpassung wird von dem Programm Z80.RSX durchgeführt, das in den 68000-Speicher geladen wird und dort solange verbleibt, bis die Reset-Taste gedrückt wird. Daher bleiben die Ladezeiten für Z80-Programme kurz.
Ein weiterer Vorteil dieses Konzepts ist, daß alle Verbesserungen des CP/M-68K (Tastatur- und Druckerpuffer, Abfrage auf Wiederholung oder Abbruch, RAM-Disk, erweitertes Random-Record-Feld sowie verbessertes Verhalten bei Diskwechsel ohne Ctrl-C) auch bei der Emulation erhalten bleiben.

Als CCP wird der CP/M-68K-Kommandoprozessor mitbenutzt. Das hat den Vorteil, daß alle Eigenschaften erhalten bleiben, etwa die Parameter-Übergabe an Submit-Dateien oder die Anzeige der aktuellen User-Nummer. Außerdem wird keine Umschaltung nötig, man kann Z80- und 68000-Betrieb mischen. Der Nachteil ist, daß Programme über ein 68000-Hilfsprogramm gestartet werden müssen: Z80.68K. Erreichbar ist die Systemerweiterung über den CP/M-68K-BDOS-Aufruf 100, der normalerweise nicht existiert. Im Register D1.L wird dazu die Adresse der Befehlszeile übergeben, wobei das in C übliche String-Format verwendet wird (keine Längenangabe, sondern Ende durch $00). Beginnt dieser String mit -S, wird ein SAVE ausgeführt, sonst wird ein Z80-Programm gestartet.

Der simulierte Z80-CCP

Die Z80-Karte kann an der Kennung $06020423 in der Slottabelle des mc-68000-Computers erkannt werden. Zur Initialisierung des Z80 wird ein Wortzugriff auf den Z80-Speicher durchgeführt, um den Z80 zurückzusetzen. Daran schließen sich einige Funktionstests an, ähnlich wie in ZRAM-TEST. Tritt hier kein Fehler auf, so wird das gewünschte Programm in den Z80-RAM ab Adresse 0100H (TPA) geladen. Wenn es zu lang ist oder Lesefehler auftreten, wird mit einer Meldung abgebrochen. Als nächstes wird die restliche Kommandozeile an den Z80 übergeben, indem sie ab Adresse 081H abgelegt wird. Als erstes Zeichen wird dabei immer ein Leerzeichen eingesetzt. Dann wird das I/O-Byte des Z80 unter Adresse 3 auf den aktuellen CP/M-68K-Wert gesetzt. Unter Adresse 4 wird als BOOT-Laufwerk die aktuelle Laufwerksnummer abgelegt. Wenn anschließend die ersten beiden Worte der Kommandozeile in zwei FCBs übersetzt sind (Adressen 05CH und 06CH), kann das Z80-Programm gestartet werden. Für dieses Programm sieht es so aus, als ob es von einem CP/M-80-CCP gestartet wurde.
Inzwischen wartet der 68000 in einer Schleife, bis er durch einen

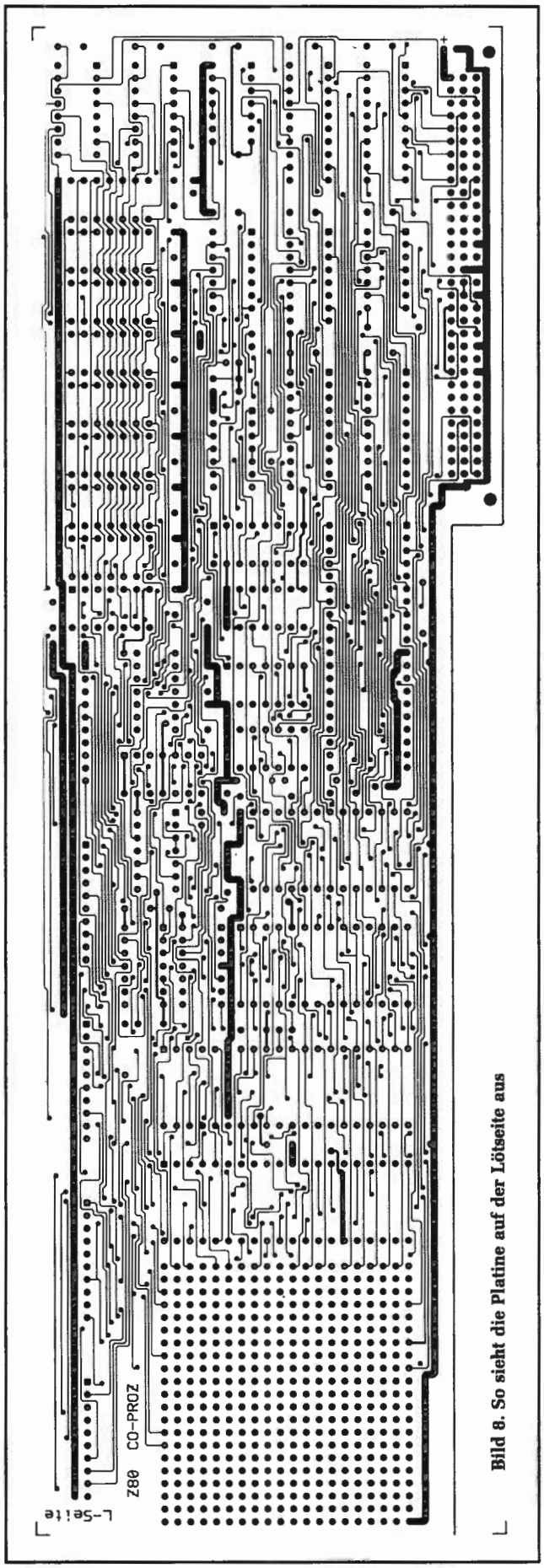


Bild 8. So sieht die Platine auf der Lötseite aus

Interrupt unterbrochen wird. Im dadurch aufgerufenen Steuerprogramm wird die gewünschte Funktion ausgeführt. Am Ende kehrt der 68000 in seine Warteschleife zurück, außer bei der BDOS-Funktion 0, bei der sich der CP/M-68K-CCP wieder meldet. Nach dem Start eines Z80-Programmes mit „Z80 -& prog.com" wird diese Schleife nicht betreten, der 68000 kehrt stattdessen sofort ins CP/M-68K zurück, und beide Prozessoren arbeiten parallel. Die im folgenden beschriebenen BDOS- und BIOS-Aufrufe können dann allerdings mit CP/M-68K-Aufrufen kollidieren, daher darf ein Z80-Programm im Parallelbetrieb keine CP/M-Aufrufe benutzen.

Das simulierte BDOS

Die BDOS-Simulation wurde auf die beim CP/M-80 vorhandenen Aufrufe beschränkt, um Kompatibilitätsprobleme zu verhindern. Außerdem wären wohl nur die zusätzlichen Aufrufe 46 (Get Disk Free Space), 48 (Flush Buffers) und 50 (Direct BIOS Call) interessant, die anderen haben nur für eine 68000-CPU eine Bedeutung. Falls sie unbedingt nötig werden, kann das Programm Z80.RSX entsprechend erweitert werden.

Die Aufrufe 15...23, 33...35 und 40 haben alle einen FCB (File Control Block) als Parameter, dessen Z80-Adresse im Registerpaar DE steht. Dieser FCB wird zunächst aus dem Z80-Speicher in den 68000-Speicher kopiert, der etwas schneller ist; außerdem verlangen die CP/M-68K-Aufrufe eine gerade Anfangsadresse, was beim Z80 nicht der Fall sein muß. Ein weiterer Grund ist, daß das Random-Record-Feld beim CP/M-68K eine andere Reihenfolge hat als bei CP/M-80, was bei der Kopie automatisch angepaßt wird. Dann wird die entsprechende Systemfunktion aufgerufen. Da sich dort etwas im FCB ändern kann, wird er am Schluß wieder zurückkopiert. Auch hier werden r0, r1, r2 wieder vertauscht. Der Wert dieser Bytes kann übrigens im Bereich 0-3FFFFH liegen. Bei Funktionen, bei denen eine Adresse in DE übergeben wird, die sich auf den Z80-Adreßraum bezieht, wird die Basisadresse der Z80-Karte addiert. Das gilt für die Funktionen 9, 10 und 26.

Viele Funktionen haben keinen oder einen Parameter, der meistens in DE oder E steht. Bei diesen Funktionen wird der Wert in das 68000-Register D1.W geladen und die Funktion direkt ausgeführt. Gegebenenfalls wird das höherwertige

Tabelle 6: Die Speicherbelegung der Z80-Karte

RAM: (0000...7FFF wenn Eprom nicht aktiv, 8000...FFFF immer)

0000...0002	jp 0fe03h	;Warmboot
0003	.dc.b 0	;I/O Byte
0004	.dc.b 0	;BOOT-Laufwerk
0005...0007	jp 0fd06h	;BDOS-Ansprung
0008...003F		;unbenutzt
0040...005B		;reserviert
005C...006B	.ds.b 16	;FCB1
006C...007B	.ds.b 16	;FCB2
007C...007F		;reserviert
0080...00FF	.ds.b 128	;Kommandozeile und DMA Puffer
0100...FCFF		;TPA 63KByte
FD00...FD05		;frei
FD06...FD3E		;BDOS-Emulation
FD3F		;frei
FD40...FDBF		;Directory buffer
FDC0...FDFF		;frei/BIOS Stack
FE00...FEBF		;BIOS-Emulation
FEC0...FECF		;noch frei
FED0...FEDF	.ds.b 16	;DPH-Kopie (BIOS select disk)
FEE0...FEEE	.ds.b 15	;DPB-Kopie (BDOS 31/BIOS select disk)
FEEF...FEFF		;unbenutzt
FF00...FFEF	.ds.b 240	;ALV-Kopie (BDOS 27 get allocation vector)
FFF0	.dc.b 0	;Startflag
FFF1	.dc.b 0	;Aufruf: 0=BDOS FF=BIOS
FFF2...FFF3	.dc.w 0	;Register BC
FFF4...FFF5	.dc.w 0	;Register DE
FFF6...FFF7	.dc.w 0	;Register HL
FFF8	.dc.b 0	;Interruptbestätigungsflag
FFF9...FFFF		;unbenutzt

EPROM: (0000...7FFF, nur wenn angeschaltet)

0000...00FF	;Start und Kopierprogramm
0100...03FF	;BDOS- und BIOS-Kopien (-> FD00++)
0400...1FFF	;unbenutzt
	;bei 27128 unbenutzt,
2000...3FFF	2764 Kopie von 0...1FFF
4000...7FFF	;Kopie von 0...3FFF

I/O: (00...FF)

00...3F		SIO
00	Datenregister A	
01	Datenregister B	
02	Controlregister A	
03	Controlregister B	
40...7F		PIO
40	Port A	
41	Port B	
42	Controlregister A	
43	Controlregister B	
80...BF		CTC
80	Kanal 0	
81	Kanal 1	
82	Kanal 2	
83	Kanal 3	
C0...FF		Interrupt und EPROM Steuerung
C0	INT aus, EPROM an	(auch nach RESET)
C1	INT aus, EPROM aus	(nach Start des Z80)
C2	INT an, EPROM an	
C3	INT an, EPROM aus	(z.B. im CP/M-Simulator)

Hinweis: durch unvollständige Dekodierung wiederholen sich die Register bei hier nicht angegebenen Adressen

Byte in D1 gelöscht, da manche CP/M-68K-Parameter 16 Bit groß sind, die bei CP/M-80 nur 8 Bit haben (z.B. Laufwerksnummer oder direkte Ein/Ausgabe).
Die Funktion 31 arbeitet etwas anders als beim CP/M-68K. Hier wird eine Kopie des DPB (Disk Parameter Block) im Z80-Speicher angelegt und die Adresse dieser Tabelle als Resultat geliefert. Beim CP/M-68K wird beim Aufruf die Adresse übergeben, in die eine Kopie gelegt werden soll. Außerdem hat sie beim 68000 ein etwas anderes Format. Im CP/M-68K gibt es keine BDOS-Funktion 27. Sie wird daher auch durch ein eigenes Programm simuliert. Dazu wird über das BIOS die Adresse des DPH (Disk Parameter Header) bestimmt. Dort findet sich einerseits die Adresse des DPB, in dem wiederum die Angabe DSM (Zahl der Blöcke minus eins) steht, aus der die Tabellenlänge bestimmt werden kann (DSM/8+1). Andererseits enthält der DPH die Adresse des 68000-ALV (Allocation Vector). Mit diesen Angaben kann die Belegungstabelle in den Z80-Speicher kopiert werden, sofern sie nicht zu groß ist (240 Byte). Als Resultat wird die Adresse der Z80-Kopie geliefert, wenn der Platz ausreicht, sonst gibt es eine Fehlermeldung mit Programmabbruch.
Das I/O-Byte wird sowohl an das CP/M-68K weitergegeben als auch an der Z80-Adresse 3 abgelegt, um die Kompatibilität zu wahren. Als Betriebssystemversion wird, wie beim CP/M-2.2 üblich, 0022H übergeben.
Resultate werden in den Registern H und L, sowie in B und A übergeben, bei 8-Bit-Resultaten sind H und B Null.
Die Error-Vektoren des CP/M-80, die bei BDOS-Fehlern angesprungen werden, können nicht unterstützt werden, da es beim CP/M-68K keine vergleichbare Möglichkeit gibt. Die Erfahrung zeigt, daß diese Vektoren aber nur selten benutzt werden.
Beachten Sie auch die Hinweise im Anhang E des CP/M-68K Programmer's Guide. Sie gelten teilweise auch für diese Simulation.

Das BIOS

Beim Z80-Programm für die BIOS-Unterstützung (*Bild 10*) fällt auf, daß es mit einer großen Sprungtabelle anfängt, wie bei CP/M-80 üblich. Jeder dieser Sprünge führt zu einem der CALL-Befehle, die alle auf ein und dieselbe Adresse springen. Wozu das Ganze?
Manche Z80-Programme verwenden direkte BIOS-Aufrufe. Dazu lesen sie aus

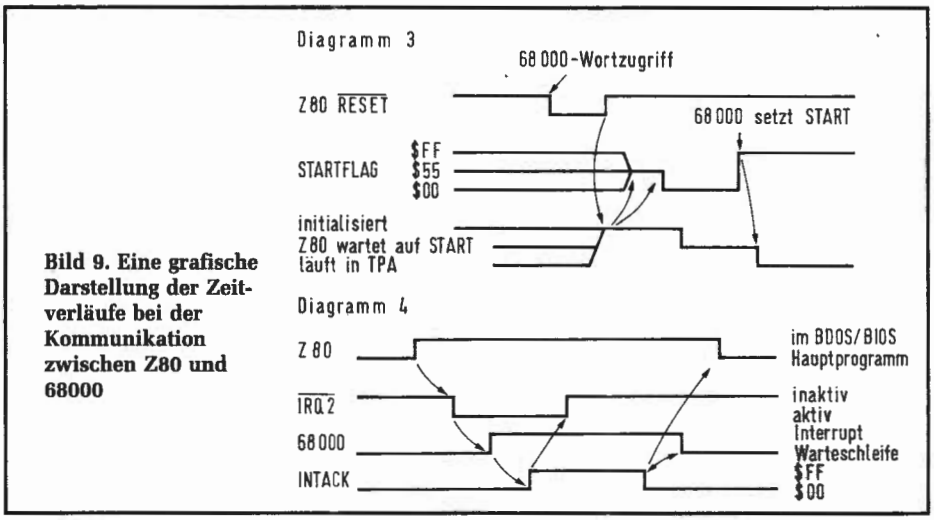


Bild 9. Eine grafische Darstellung der Zeitverläufe bei der Kommunikation zwischen Z80 und 68000

Adressen 0001H und 0002H die Adresse des BIOS-Warmstarts, nehmen an, daß dort die Sprungabelle steht, und holen sich die Sprungziele der benötigten BIOS-Aufrufe, statt die Adresse des JP-Befehls zu nehmen. Das hat den Vorteil, einen unnötigen Sprung einzusparen, für ein BIOS bedeutet es, daß unter jeder dieser Adressen ein ausführbares Programm stehen muß.
Wird einer dieser CALLs angesprungen, legt er die Adresse des nächsten Befehls auf den Stack, das gemeinsame BIOS-Steuerprogramm kann aus dieser Rückkehradresse die BIOS-Aufruf-Nummer bestimmen. Benutzt wird sie immer ohne expliziten Rücksprung. Damit diese Berechnung etwas einfacher wird, liegen alle CALL-Befehle auf geraden Adressen.
Nun wieder zum 68000-Steuerprogramm: wie beim BDOS werden auch beim BIOS viele Systemaufrufe direkt an das CP/M-68K weitergeleitet. Das geschieht über den BDOS-Aufruf 50 (Direct BIOS Call), damit automatisch alle Register gerettet werden. Wenn Argumente übergeben werden, stehen sie in der Z80-Registerkopie BC, Ergebnisse werden im Akkumulator abgelegt.
CP/M-68K erwartet beim BIOS-Aufruf 9 eine Angabe, ob das Laufwerk zum ersten Mal selektiert wird, damit eine automatische Formaterkennung nicht jedesmal durchgeführt werden muß. Da das CP/M-80 diese Information nicht lie-

```
0001 0000                       ;
0002 0000                       ; Z80-COPROCESSOR EPROM-Inhalt
0003 0000                       ;
0004 0000                       ; H.N.Schaller
0005 0000                       ;
0006 0000                       ; 26.06.1986
0007 0000                       ;
0008 0000                       .abs    0
0009 0000 06 00                 ld      b,0             ;auf SIO- und PIO-Init. warten
0010 0002 10 FE         INIT:   djnz    INIT            ;Warteschleife
0011 0004 21 00 01              ld      hl,COPY         ;Adresse des zu kopierenden Programms
0012 0007 11 00 FD              ld      de,BDOS         ;Zieladresse
0013 000A 01 F9 02              ld      bc,INTACK-BDOS+1 ;Gesamtgröße
0014 000D ED B0                 ldir                    ;kopieren
0015 000F C3 A0 FE              jp      ENTRY           ;und anspringen
0016 0012 5A 38 30 20           .db     'Z80 COPROCESSOR 26.6.86 HNS'
0016 0016 43 4F 50 52
0016 001A 4F 43 45 53
0016 001E 53 4F 52 20
0016 0022 32 36 2E 36
0016 0026 2E 38 36 20
0016 002A 48 4E 53
0017 0100                       .abs    100H            ;Kopie beginnt hier
0018 0100               COPY:                           ;Adresse im EPROM
0019 0100                       .move   0FD00H          ;BDOS läuft bei 0FD00H, Kopie: 100H
0020 FD00:00 00 00 00 BDOS:     .dw     0,0,0           ;Füllbytes
0020 0104 00 00
0021 0106 C3 11 FD              jp      BDOS1           ;überspringen
0022 0109 00 00 00 00           .dw     0,0,0,0         ;ERROR-Vektoren (unbenutzt)
```

Bild 10. Mit diesem Hilfsprogramm im EPROM laufen auf der Z80-Karte CP/M-2.2-Programme, Systemaufrufe werden dabei vom CP/M-68k behandelt

```
0022 010D 00 00 00 00
0023 FD11:ED 43 F2 FF BDOS1: ld      (RBC),bc    ;BC und DE retten (BDOS-Parameter)
0024 0115 ED 53 F4 FF        ld      (RDE),de
0025 0119 AF                 xor     a           ;A=0
0026 FD1A:32 F1 FF    BDOS2: ld      (FLAG),a    ;Aufruf setzen
0027 011D D3 C3              out     (0C3H),a    ;Interrupt an (EPROM ist aus)
0028 011F 21 00 00           ld      hl,0        ;Wartezeit ca 0.5sec
0029 FD22:3A F8 FF    BDOS3: ld      a,(INTACK)  ;hat der 68000 reagiert?
0030 0125 A7                 and     a           ;0 heißt ja
0031 0126 20 09              jr      nz,BDOS4    ;ja
0032 0128 2B                 dec     hl
0033 0129 7C                 ld      a,h
0034 012A B5                 or      l           ;Zähler = 0?
0035 012B 20 F5              jr      nz,BDOS3    ;nein, weiter warten
0036 012D D3 C1              out     (0C1H),a    ;Interrupt aus, 68000 reagiert nicht
0037 FD2F:18 FE       STOP:  jr      STOP        ;Endlosschleife bis Reset
0038 FD31:D3 C1       BDOS4: out     (0C1H),a    ;ok, Interrupt zurücknehmen
0039 FD33:3A F8 FF    BDOS5: ld      a,(INTACK)  ;Interrupt verarbeitet?
0040 0136 A7                 and     a
0041 0137 20 FA              jr      nz,BDOS5    ;nein, noch nicht
0042 0139 2A F6 FF           ld      hl,(RHL)    ;Ergebnis abholen
0043 013C 44                 ld      b,h
0044 013D 7D                 ld      a,l         ;für kurzes Ergebnis
0045 013E C9                 ret                 ;BDOS/BIOS-Aufruf beendet
0046 013F                    ;
0047 0200                    .abs    200H        ;Adresse im EPROM
0048 0200                    .move   0FE00H      ;absolute Adresse
0049 FE00:            STACK:                     ;Stack vor BIOS
0050 0200 C3 3C FE           jp      CBOOT       ;Sprungtabelle der BIOS-Aufrufe
0051 FE03:C3 40 FE    wboot: jp      WBOOT
0052 0206 C3 44 FE           jp      CSTAT
0053 0209 C3 48 FE           jp      CIN
0054 020C C3 4C FE           jp      COUT
0055 020F C3 50 FE           jp      LOUT
0056 0212 C3 54 FE           jp      PUN
0057 0215 C3 58 FE           jp      RDR
0058 0218 C3 5C FE           jp      HOME
0059 021B C3 60 FE           jp      DISK
0060 021E C3 64 FE           jp      TRACK
0061 0221 C3 68 FE           jp      SECTOR
0062 0224 C3 6C FE           jp      DMA
0063 0227 C3 70 FE           jp      READ
0064 022A C3 74 FE           jp      WRITE
0065 022D C3 78 FE           jp      LSTAT
0066 0230 C3 7C FE           jp      TRANS
0067 0233 C3 80 FE           jp      USER1
0068 0236 C3 84 FE           jp      USER2
0069 0239 C3 88 FE           jp      USER3
0070 023C                    ;hier weitere BIOS-Aufrufe anfügen
0071 FE3C:CD 8C FE    CBOOT: call    BIOS        ;Rückkehradresse auf den Stack
0072 FE3F:00          RRR:   nop                 ;dieser Wert auf Stack, wenn CBOOT
0073 FE40:CD 8C FE    WBOOT: call    BIOS
0074 0243 00                 nop                 ;4 Byte/Aufruf vermeidet Div. mit 3
0075 FE44:CD 8C FE    CSTAT: call    BIOS
0076 0247 00                 nop
0077 FE48:CD 8C FE    CIN:   call    BIOS
0078 024B 00                 nop
0079 FE4C:CD 8C FE    COUT:  call    BIOS
0080 024F 00                 nop
0081 FE50:CD 8C FE    LOUT:  call    BIOS
0082 0253 00                 nop
0083 FE54:CD 8C FE    PUN:   call    BIOS
0084 0257 00                 nop
0085 FE58:CD 8C FE    RDR:   call    BIOS
0086 025B 00                 nop
0087 FE5C:CD 8C FE    HOME:  call    BIOS
0088 025F 00                 nop
0089 FE60:CD 8C FE    DISK:  call    BIOS
0090 0263 00                 nop
0091 FE64:CD 8C FE    TRACK: call    BIOS
0092 0267 00                 nop
0093 FE68:CD 8C FE    SECTOR:call    BIOS
0094 026B 00                 nop
0095 FE6C:CD 8C FE    DMA:   call    BIOS
0096 026F 00                 nop
0097 FE70:CD 8C FE    READ:  call    BIOS
0098 0273 00                 nop
0099 FE74:CD 8C FE    WRITE: call    BIOS
0100 0277 00                 nop
0101 FE78:CD 8C FE    LSTAT: call    BIOS
0102 027B 00                 nop
0103 FE7C:CD 8C FE    TRANS: call    BIOS
0104 027F 00                 nop
0105 FE80:CD 8C FE    USER1: call    BIOS
0106 0283 00                 nop
0107 FE84:CD 8C FE    USER2: call    BIOS
0108 0287 00                 nop
0109 FE88:CD 8C FE    USER3: call    BIOS
0110 028B 00                 nop
0111 028C                    ;hier weitere Aufrufe einfügen
0112 FE8C:ED 43 F2 FF BIOS:  ld      (RBC),bc    ;Parameter abspeichern
0113 0290 ED 53 F4 FF        ld      (RDE),de
0114 0294 E1                 pop     hl          ;Rücksprungadresse von call BIOS
0115 0295 7D                 ld      a,l
0116 0296 D6 3F              sub     RRR         ;Aufrufsnummer erzeugen (k*4)
0117 0298 32 F6 FF           ld      (RHL),a     ;in H-Register speichern
0118 029B 3E FF              ld      a,-1        ;ist BIOS-Aufruf
0119 029D C3 1A FD           jp      BDOS2       ;Aufruf ausführen
0120 02A0                    ;hierher wird nach Reset gesprungen
0121 FEA0:D3 C1       ENTRY: out     (0C1H),a    ;EPROM abschalten (kein Interrupt)
0122 02A2 21 C0 FE           ld      hl,JMPTAB   ;ist zwar schon Kopie....
0123 02A5 11 00 00           ld      de,0        ;Sprünge auf BIOS und BDOS setzen
0124 02A8 01 08 00           ld      bc,8        ;Adressen 0..7 im RAM ändern
0125 02AB ED B0              ldir                ;Kopieren
0126 02AD 31 00 FE           ld      sp,STACK    ;Stack zwischen BDOS und BIOS legen
0127 02B0 AF                 xor     a
0128 02B1 32 F0 FF           ld      (START),a   ;Z80 läuft jetzt
0129 FEB4:3A F0 FF    ENT1:  ld      a,(START)
0130 02B7 A7                 and     a
0131 02B8 28 FA              jr      z,ENT1      ;auf Startsignal vom 68000 warten
0132 02BA CD 00 01           call    TPA         ;und Programm im RAM aufrufen
0133 02BD C3 03 FE           jp      wboot       ;Warmboot, Anwenderprogramm beendet
0134 02C0                    ;Sprünge in BIOS und BDOS (können überschrieben werden!)
0135 02C0                    ;diese Werte werden nach 0..7 kopiert
0136 FEC0:C3 03 FE    JMPTAB: jp     wboot       ;Kaltstart (RST 0)
0137 02C3 00                 .db     0           ;I/O-Byte (Wert irrelevant)
0138 02C4 00                 .db     0           ;Bootdrive
0139 02C5 C3 06 FD           jp      BDOS+6      ;BDOS-Aufruf
0140 02C8                    ;gemeinsamer Speicher für Parameterübergabe
0141 02C8                    ; DPH .ds 16
0142 02C8                    ; DPB .ds 15
0143 02C8                    ; ALV .ds 240
0144 03F0                    .abs    3F0H        ;wird aus dem EPROM initialisiert
0145 03F0                    .move   0FFF0H
0146 FFF0:55          START: .db     055H        ;Startsignal wenn <> 0
0147 FFF1:01          FLAG:  .db     01H         ;0=BDOS FF=BIOS
0148 FFF2:02 03       RBC:   .dw     0302H       ;BC-Register, C=BDOS-Funktionsnummer
0149 FFF4:04 05       RDE:   .dw     0504H       ;DE-Register
0150 FFF6:06 07       RHL:   .dw     0706H       ;HL-Register, H=BIOS-Funktionsnummer
0151 FFF8:00          INTACK:.db     0           ;Interrupt Bestätigung
0152 0100:            TPA:   =       100H        ;wird durch 'START' aufgerufen
0153 03F9                    .end
```

fern kann, wird die Formaterkennung einfach jedesmal verlangt. Die Adresse des zu diesem Laufwerk gehörenden DPH wird – ebenso wie die Laufwerksnummer – abgespeichert, damit bei einem Sektorumsetzaufruf (BIOS 16) auf die 68000-Tabelle zugegriffen werden kann. Als Ergebnis erhält der Z80 die Adresse des DPH, der vorher an einer festen Position im Z80-Speicher angelegt wurde. Dort stehen verschiedene Parameter und Zeiger. Die Adresse der Sektorumsetztabelle kann dort nicht abgelegt werden, da sie nicht in den Z80-Speicher kopiert wird, weil die Länge einerseits kaum zu bestimmen ist, andererseits für jedes Laufwerk eine eigene Kopie angelegt werden müßte.
Statt dessen wird eine Pseudoadresse übergeben, die aus einer Konstanten und der Laufwerksnummer besteht. Im allgemeinen lesen CP/M-80-Programme diese Adresse aus dem DPH, nur um sie an den Umsetzaufruf zu übergeben. Allerdings darf dieser Wert nicht zwischengespeichert werden, um auf mehrere Laufwerke zugreifen zu können. Das geht nicht, weil alle solchen Tabellen auf die gleiche Adresse geschrieben werden. Sinnvoll wäre das auch nicht, da ohnehin nur auf die Sektoren des zuletzt selektierten Laufwerks zugegriffen werden kann. Von diesem stammt auch der aktuelle DPH. Der Aufruf 16 prüft dann, ob die korrekte Pseudoadresse zurückgegeben wurde, und nimmt statt dessen die vom letzten Diskselekt bekannte 68000-Tabellenadresse.
Die Aufrufe, mit denen die Spur und der Sektor angewählt werden, verwenden hier auch Wortparameter, die aber bei den Sektornummern im BIOS des mc-68000-Computers nicht ausgenutzt werden. Die Umsetzung der BIOS-DMA-Adresse läuft genauso wie beim BDOS. Zusätzlich wird aber die BDOS-DMA-Adresse verändert, da beim CP/M-80 nur ein gemeinsamer Zwischenspeicher im BIOS existiert – im CP/M-68K gibt es jedoch auch einen im BDOS, der unabhängig verändert werden kann.

BIOS-Erweiterungen

Die BIOS-Aufrufe 18...22 können nicht angesprochen werden, da sie entweder überflüssig (I/O-Byte ändern/abfragen) oder 68000-spezifisch sind. Dafür bietet die Simulation drei weitere Sprünge ins BIOS, über die eigene Erweiterungen aufgerufen werden können. Wenn man diese Funktionen benutzen will, muß das unterstützende Programm Z80.RSX ebenfalls erweitert werden. Dort gibt es eine Sprungtabelle für alle BIOS-Aufrufe, in der die Adresse des Erweiterungsprogramms eingetragen wird. Bei einem entsprechenden BIOS-Aufruf wird es dann aufgerufen.
Das Register A6 enthält dabei die Adresse der Z80-Karte plus $8000. Mit $7ff2(A6) erreichet man z. B. die Adresse 0FFF2H im Z80-Adreßraum, dort steht der Inhalt des C-Registers, $8080(A6) entspricht 80H, also der Kommandozeile. Für Verwirrung sorgt dabei, daß beim Z80 im Gegensatz zum 68000 das niederwertigere Byte unter der niedrigeren Adresse steht. So können die Registerinhalte ausgelesen (BC und DE können Parameter beinhalten) und Resultate in HL übergeben (H=B und L=A) werden. Was in den Registern steht, bleibt dem Anwenderprogramm überlassen, wenn es Adressen im Z80-Speicher sind, muß A6 addiert und $8000 subtrahiert werden, damit sich die 68000-Adresse ergibt. Bei der Orientierung hilft ein Blick auf die bisherigen Programmteile (Bild 10) und die Aufteilung des Z80-Speichers (*Tabelle 6*).
Eine andere Möglichkeit, das BIOS (oder gar das BDOS) zu erweitern, besteht darin, auch das Programm im BOOT-EPROM zu ändern. Falls die Bereiche für BIOS und BDOS nicht ausreichen, setzt man die Anfangsadressen entsprechend herab. Dadurch wird der TPA-Bereich kleiner, das Programm Z80.RSX erkennt diese Änderung automatisch. Denkbar wäre zum Beispiel ein erweitertes BIOS oder gar ein vollständiges BDOS. Auch lassen sich die BIOS-Aufrufe Punch und Reader beispielsweise auf einen der SIO-Kanäle leiten, für Z80-Programme wäre dann ein unabhängiger, asynchroner Kanal vorhanden.

Die Programme

Z80INIT.68K trägt die Z80-Karte in dem als Parameter angegebenen Platz der Slottabelle (sinnvoller Bereich 2 bis 7) ein und lädt Z80.RSX an das obere Speicherende. Z80.RSX ist das Treiberprogramm, das die Datenübergabe zwischen Z80 und 68000 steuert. Z80.68K startet COM-Dateien, SAVE.68K stellt den im CP/M-80-CCP vorhandenen gleichnamigen Befehl zur Verfügung. ZRAM-TEST.68K ist das schon oben beschriebene RAM-Testprogramm.
Am besten kopiert man die oben genannten Programme zusammen mit CONFIG.68K auf eine leere Diskette. Dann ändert man mit CONFIG den Autostart auf „Z80INIT 3", wenn die Z80-Karte in Steckplatz 3 steckt. Soll z.B. eine Uhrenkarte gemeinsam installiert werden, müssen die dazu benötigten Programme kopiert werden. Der Autostart lautet dann z. B. „Z80INIT 3 ! UHR" (zwei Befehle in einer CCP-Zeile) und es werden beide Initialisierungsprogramme ausgeführt. Die Kommandozeile muß ausnahmsweise in Großbuchstaben geschrieben sein, sonst werden die Dateien nicht erkannt.
Werden nun CP/M-80-Programme auf diese Diskette kopiert, lassen sie sich sofort ausführen. Um z. B. DDTZ mit TEST.COM als Argument zu starten, gibt man „z80 ddtz test.com" ein. Bei einem CP/M-80 würde nur das „z80" weggelassen. Der Debugger lädt test.com, diese Datei läßt sich nun anschauen und ändern. Das Abspeichern der geänderten Datei erledigt SAVE: Nach Verlassen von DDTZ behält der Z80-Speicher seinen gesamten Inhalt.
Mit „save 12 test1.com" werden die ersten 12 Blöcke zu 256 Byte in die Datei test1.com abgespeichert, wie man es von einem CP/M-80 erwarten würden.
Als Alternative kann jedoch auch die absolute Speicherobergrenze angegeben werden, z.B. „save $144b test2.com". Dann wird sichergestellt, daß von $100 bis mindestens $144b alles aufgezeichnet wird, in diesem Beispiel ist das bis $147f.
Als letzte Vereinfachung bietet CP/M-68K Parameter bei Submit-Dateien an. Wenn in einer Datei „DDTZ.SUB" folgendes steht: „z80 ddtz $1" (ohne Anführungszeichen), dann kann man im vorangegangenen Beispiel auch „ddtz test.com" eingeben. Dieser Befehl führt die Submit-Datei aus, $1 wird dabei durch test.com ersetzt. Der entstandene Befehl ist mit dem in der ersten Version identisch. Diese Methode läßt sich auf alle Z80-Programme anwenden, wobei eventuell weitere Parameter ($2 $3 $4 usw.) nötig sind.
Interessant ist sicher auch die Möglichkeit, einen guten CP/M-80-Texteditor, die Hilfsprogramme und einen 68000-Compiler auf eine Diskette zu legen. Der Editor läßt sich über eine Submitdatei und Z80.68K laden und starten, während der 68000-Compiler ein direkt ausführbares Programm ist, und daher vom CP/M-68K direkt geladen und gestartet wird.
Bei der Betrachtung des Wrap-Feldes (*Bild 11*), auf dem sich ja einige Schnittstellen aufbauen lassen, fallen dem engagierten Computeristen sicher sofort etliche Aufgaben ein, die ein intelligenter, eigenständiger Subrechner hier übernehmen könnte – aber Vorsicht: Mehr als sechs dieser Karten lassen sich im mc-68000-Computer nicht unterbringen...

Thomas Schlenger-Klink

Basic-EMUF mit LCD und Tasten

Bei vielen Anwendungen von Einplatinen-Computern ist die Ein-/Ausgabe von Daten während des Betriebs notwendig, so auch beim Basic-EMUF. Die Programmierung einer solchen Ein-Ausgabe-Einheit ist bei Realisierung in Maschinensprache relativ umfangreich und zeitaufwendig. Die hier vorgestellte Lösung LCDKEY ist einfach zu handhaben, da die Programmierung mit den Basic-Befehlen „PRINT" und „INPUT" erfolgt. Die Daten können dabei alphanumerisch angezeigt und eingegeben werden.

Die LCDKEY-Baugruppe wurde in erster Linie als Ein-/Ausgabe-Baugruppe für den Basic-EMUF entwickelt. Der Anschluß an andere Computer ist jedoch ebenfalls möglich. Die Ansteuerung der Tastatur und des LC-Moduls erfolgt über einen PIO-Baustein 8255. Dieser Baustein stellt 24 TTL Ein-Ausgabeleitungen zur Verfügung. Damit kann die Platine auch an andere Computer angeschlossen werden (z.B. 8085, Z80 etc.). Die Anzeige besteht aus einem fertig montierten LC-Modul, das in den verschiedensten Ausführungsformen erhältlich ist (mit und ohne Beleuchtung, 16-40 stellig). Die Taster des Tastenfeldes sind so konstruiert, daß sie mit einlegbaren Papierschildern sehr einfach und dauerhaft selbst beschriftet werden können, es sind jedoch auch fertig beschriftete Tastenkappen lieferbar.

Schaltungsbeschreibung

Bild 1 zeigt das Schaltbild des gesamten LCDKEY-Moduls. Das Anzeigemodul enthält bereits die gesamte Ansteuerungselektronik für das Display. Auf der Platine kann eine Anzeige mit zwei Zeilen zu je 16 Stellen montiert werden. Das Platinenformat wurde so gewählt, daß sich das LCDKEY-Modul sehr gut in einem 19-Zoll Einschubsystem unterbringen läßt. Andere Module (20, 32, 40 Stellen) werden über ein Kabel an die Grundplatine angeschlossen, da die unterschiedlichen Module meist die gleiche Ansteuerungselektronik enthalten. Das Anzeigemodul stellt den kompletten ASCII-Zeichensatz dar. Darüberhinaus verfügt die Anzeige über einen selbstprogrammierbaren Zeichengenerator zur Anzeige selbstentworfener Zeichen (z.B. Umlaute). Das LC-Modul ist über einen PIO-Baustein 8255 an den Datenbus des Basic-EMUF angeschlossen (über PIO-Stecker ST4). Dadurch werden auf dem Basic-EMUF keine I/O-Leitungen belegt. Der Datenbus der Anzeige ist an Port A des 8255 angeschlossen, die restlichen Steuerleitungen (RS, E, R/W) an Port C.

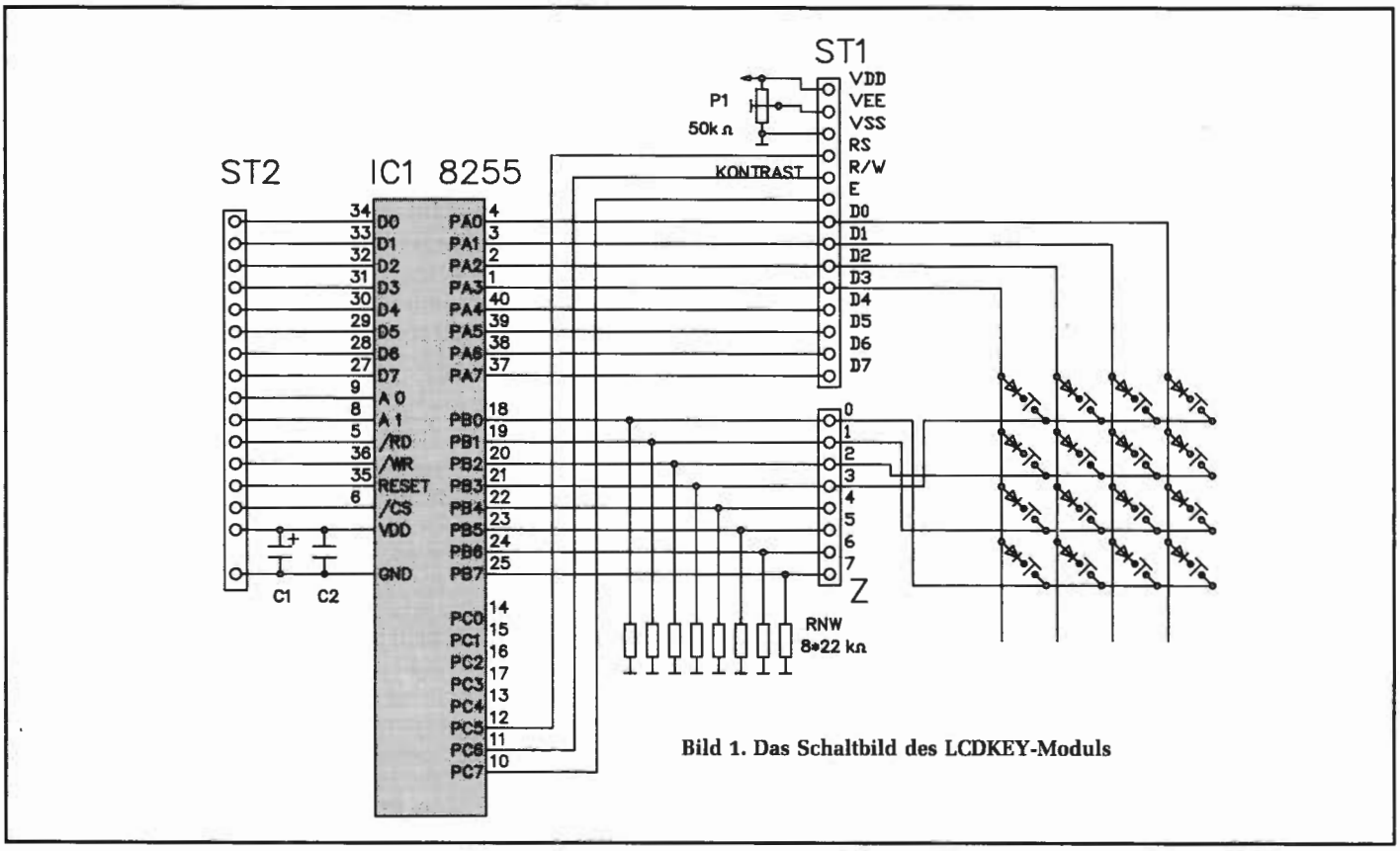


Bild 1. Das Schaltbild des LCDKEY-Moduls

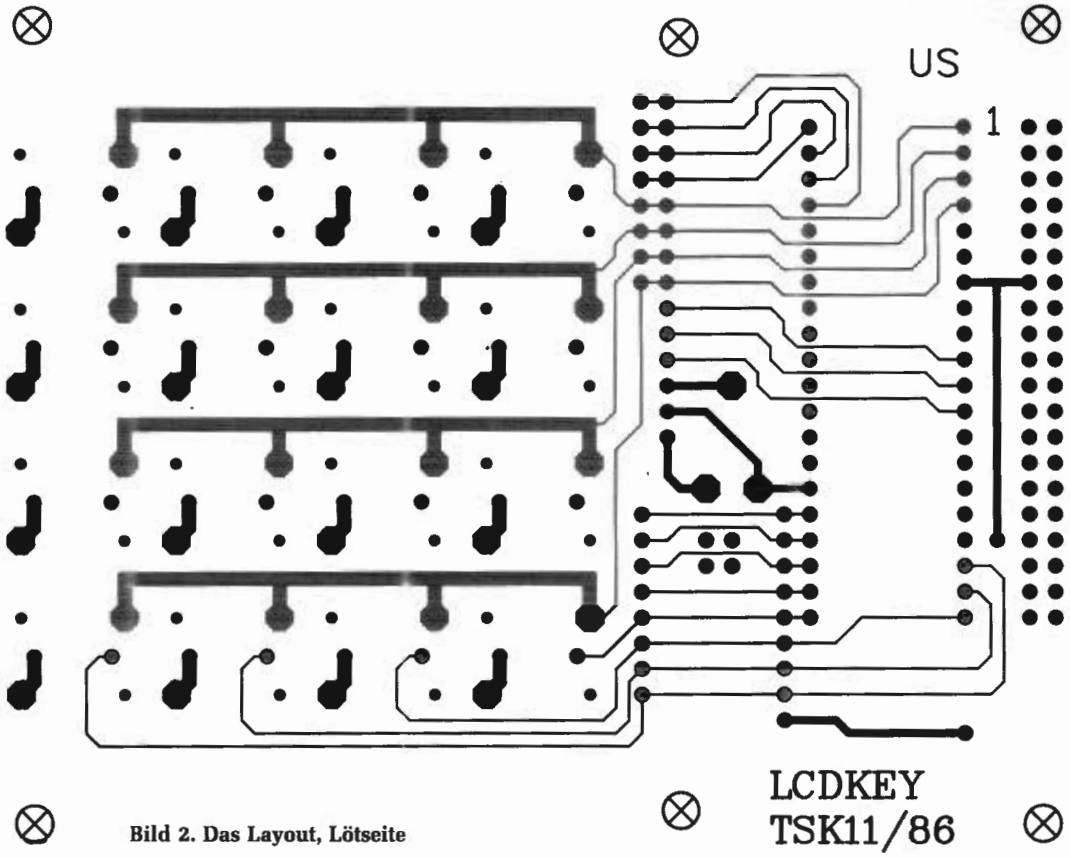


Bild 2. Das Layout, Lötseite

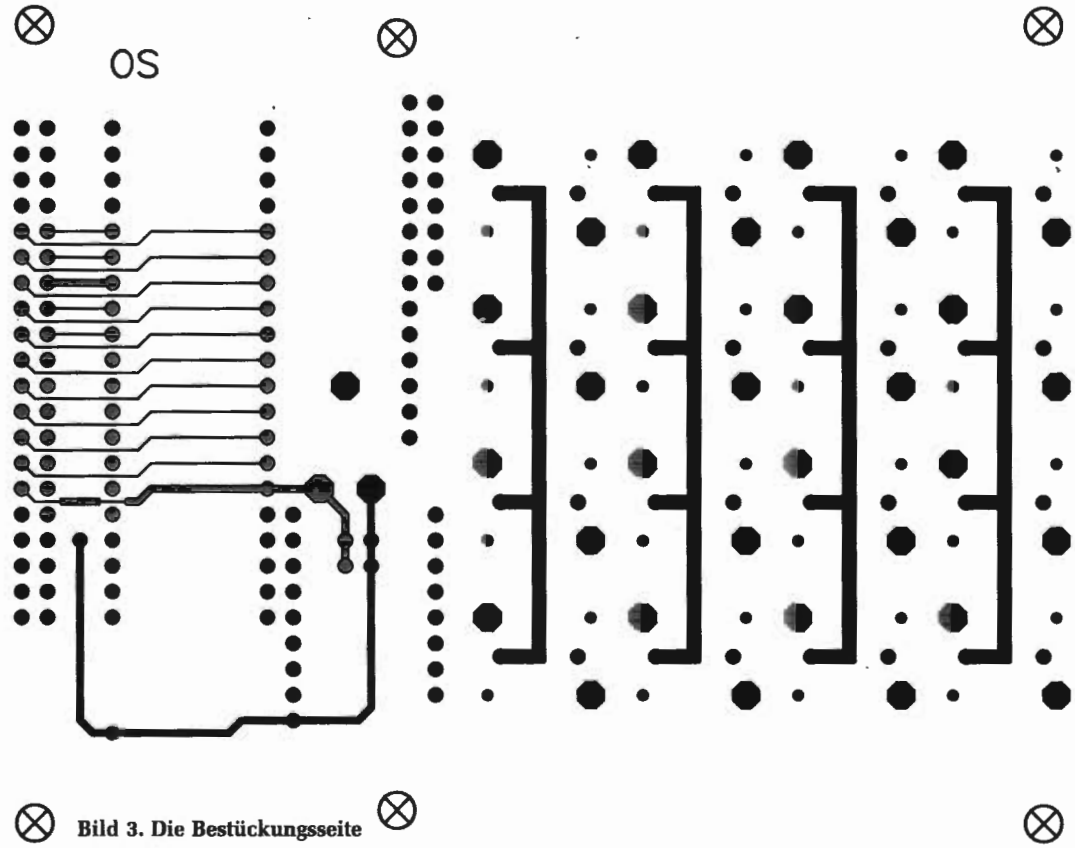


Bild 3. Die Bestückungsseite

Für den PIO-Baustein sollte nach Möglichkeit eine CMOS-Version (82C55, 71055) eingesetzt werden, um den Stromverbrauch so gering wie möglich zu halten. Er beträgt dann nur einige Milliampere (ca. 3 mA).
Die Ansteuerung der Tastenmatrix erfolgt mit Port A (Ausgänge) und Port B (Eingänge). Zur Entkoppelung der einzelnen Tasten untereinander dient jeweils eine Diode an jedem Tastenkontakt. Damit ist es möglich auch Tastenkombinationen von zwei und mehr Tasten auszuwerten (z.B. SHIFT-Taste).
Die Abfrage erfolgt in der Weise, das am Port A jeweils eine Leitung auf „HIGH" gesetzt wird und dann am Port B geprüft wird, ob irgendein Eingang ebenfalls auf „HIGH" liegt.

Aufbau

In *Bild 2 und 3* ist das Layout der zweiseitigen Platine angegeben. *Bild 4* zeigt den Bestückungsplan der Baugruppe. Zur Verwendung einer eigenen Tastatur sind entsprechende Kontakte auf der Platine vorgesehen. Das Tastenfeld kann dazu abgetrennt werden. Bei Verwendung des vorgesehenen Tastenfeldes muß darauf geachtet werden, daß zuerst die Dioden unter den Tastern , dann erst die Taster selbst eingebaut werden. Die restlichen Bauteile nach Stückliste werden von der Bestückungsseite der Platine eingebaut. Vor der Montage des Displays ist die Platine sorgfältig auf Löt- und Bestückungsfehler zu überprüfen. Das LC-Modul wird mechanisch mit kleinen Abstandsbolzen befestigt, erst danach erfolgt der elektrische Anschluß.

Inbetriebnahme

Nach genauer Überprüfung auf Fehler wird die LCDKEY Baugruppe an den Basic-EMUF mit einem 40 poligen Flachbandkabel am PIO-Stecker (ST4) angeschlossen. Die Länge dieses Kabels sollte 30 cm nicht übersteigen. Ein EPROM mit dem angebenen Treiberprogramm muß im Assembler-EPROM-Sokkel (IC 8) des Basic-EMUF eingesteckt sein. Nach dem Einschalten mit angeschlossenem LCDKEY sollte sich der Basic-EMUF wie gewohnt am Terminal melden. Der Kontrasteinsteller auf der Rückseite der Platine wird jetzt so eingestellt, daß die obere Reihe der Anzeige dunkle Kästchen anzeigt (fast rechter Anschlag von P1). Nach der Initialisierung mit CALL 1 ist jetzt ein blinkender Cursor in der oberen linken Ecke sichtbar. Mit dem Befehl „UO1" kann nun die Ausgabe auf das Display umgeleitet werden. Mit dem Befehl „UO0" kann jederzeit wieder auf das Terminal zurückgeschaltet werden.

Universal Input/Output Programm

Ein Bausatz mit Anleitung und universellem Input-/Output-Programm ist beim Elektronikladen in Detmold erhältlich. Die mc-Redaktion verschickt das Listing von UIO auf Anfrage.
Das Ansteuerprogramm für die LCDKEY-Baugruppe belegt nur einige hundert Byte des Eproms. Es wurden noch einige andere Unterprogramme im EPROM untergebracht. Diese dienen hauptsächlich der Ansteuerung verschiedener Peripheriebaugruppen des Basic-EMUFs. Weiterhin wurde ein Unterprogramm hinzugefügt, das die Funktion VALUE (Umwandlung eines Strings in eine Fließkommazahl) durchführt.
Für die jeweiligen Funktionen sind in *Bild 5* verschiedene einfache Testprogramme angegeben, die die Verwendung dieser kleinen Hilfsprogramme zeigen.

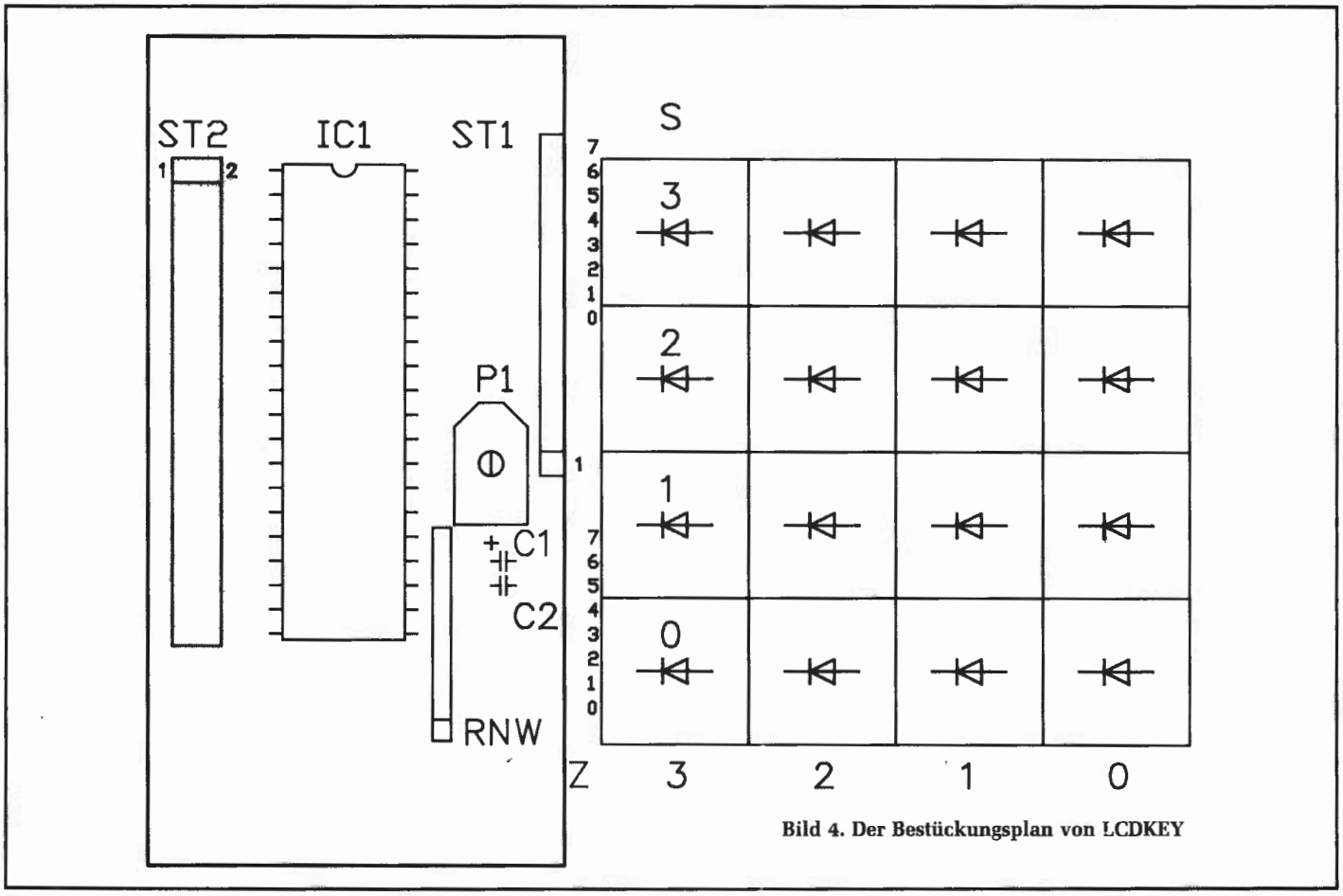


Bild 4. Der Bestückungsplan von LCDKEY

```
Aufruf    NAME      Funtion
CALL 0    DORES     Führt einen absoluten Sprung nach 0h aus.
                    Entspricht in etwa einem Software Reset
CALL 1    LCDINI    Initialisierung LCDKEY
CALL 2    LCDDAT    Ausgabe von [TOS] an das Daten-Register
                    des LCD-Moduls. Dient zur Ansteuerung des
                    programmierbaren Zeichengenerators usw.
CALL 3    LCDCON    Ausgabe von [TOS] an das Control-Register
                    des LC-Moduls. Dient hauptsächlich zur
                    freien CURSOR-Positionierung
CALL 4    ADC       Messen eines AD-Kanals. Kanal in [TOS].
                    Ergebnis wird in [TOS] zurückgeliefert
CALL 5    READTM    Einlesen der Uhrzeit nach $(0).
                    Strings müssen mit min. 17 Zeichen definiert
                    sein (z.B. STRING 100,20)
CALL 6    RDNBAS    Lesen eines 4-Bit Wertes (Nibble) der Uhr RTC58321.
                    Aufruf mit [TOS]=Adresse des Nibbles.
                    Rückkehr mit [TOS]=Daten des Nibbles
CALL 7    WRNBAS    Schreiben eines 4-Bit Wertes (Nibble) der RTC58321.
                    Aufruf mit [TOS-1] = zu schreibender Wert
                               [TOS]   = Adresse des Nibbles
CALL 8    VALUE     Wandelt $(0) in eine Fließkommazahl
                    Rückkehr [TOS] = Status, bei Fehler<>0
                    [TOS-1] = Fließkommazahl wenn Status=0
CALL 9    FORINI    Initialisierung der Parameter für die Drucker-
                    schnittstelle mit [TOS]. Freigabe des PRINT@,LIST@
                    Treiberprogramms
```

Programmbeispiele:

```
10    REM TESTPROGRAMM ADC,LCD
20    PRINT "BEENDEN DURCH BELIEBIGEN TASTENDRUCK"
30    CALL 1 :  UO 1 :  REM INITIALISIERUNG UND UMSCHALTEN AUF LCDKEY
40    PRINT CHR(5) :  REM CURSOR AUS
50    PUSH 0 :  CALL 4 :  REM MESSEN KANAL 0
60    POP W :  REM W = MESSWERT
70    PUSH 80H :  CALL 3 :  REM CURSOR 1.ZEICHEN 1.ZEILE POS.
80    PRINT W :  REM AUSGABE DES MESSWERTES
90    FOR I=1 TO 100 :  NEXT I :  REM WARTESCHLEIFE
100   A=GET :  IF A=0 THEN 50
110   UO 0

10    REM TESTPROGRAMM UHR
20    STRING 106,20 :  REM STRINGRESERVIERUNG 5*20 BYTE
30    PRINT "ZUM STELLEN DER UHR MUSS JUMPER J5 GESCHLOSSEN SEIN"
40    PRINT
50    INPUT "WELCHES NIBBLE SOLL BESCHRIEBEN WERDEN ? ",AD
60    PUSH AD :  CALL 6 :  REM LESEN RTC
70    POP DA :  REM WERT HOLEN
80    PRINT "MOMENTANER WERT =",DA,
90    INPUT "NEUER WERT ? ",DA
100   PUSH DA,AD :  CALL 7 :  REM SCHREIBEN DES NEUEN WERTES
110   CALL 5 :  PRINT $(0) :  REM AUSGABE DER UHRZEIT
120   PRINT "WEITERMACHEN (J/N) ? ",
130   A=GET :  IF A=0 THEN 130 :  REM WARTEN AUF TASTENDRUCK
140   IF A=ASC(J) THEN 40

10    REM TESTPROGRAMM UHR,LCD
20    STRING 106,20 :  REM STRINGRESERVIERUNG 5*20 BYTE
30    PRINT "BEENDEN DURCH BELIEBIGEN TASTENDRUCK"
40    CALL 1 :  UO 1 :  REM INITIALISIERUNG UND UMSCHALTEN AUF LCDKEY
50    PRINT CHR(5) :  REM CURSOR AUS
60    CALL 5 :  REM LESEN DER UHRZEIT NACH $(0)
70    PUSH 80H :  CALL 3 :  REM CURSOR 1.ZEICHEN 1.ZEILE POS.
80    FOR I=1 TO 9 :  PRINT CHR(ASC($(0),I)), :  NEXT I :  REM DATUM
90    PUSH 0C0H :  CALL 3 :  REM CURSOR 1.ZEICHEN 2.ZEILE POS.
100   FOR I=10 TO 18 :  PRINT CHR(ASC($(0),I)), :  NEXT I :  REM ZEIT
110   A=GET :  IF A=0 THEN 50 :  REM WARTEN AUF TASTENDRUCK
120   UO 0

10    REM TESTPROGRAMM TASTENFELD
20    STRING 106,20 :  REM RESERVIERUNG STRINGSPEICHERPLATZ
30    CALL 1 :  REM INITIALISIERUNG LCDKEY
40    PRINT "GEBEN SIE EINEN STRING AM 16ER-TASTENFELD EIN"
50    UI 1 :  REM UMSCHALTEN TASTENFELD
60    INPUT $(0)
70    UI 0 :  REM ZURUECKSCHALTEN AUF TERMINAL
80    PRINT $(0)

10    REM TEST VALUE FUNKTION
20    STRING 100,10
30    INPUT $(0)
40    CALL 8
50    POP ST
60    IF ST<>0 THEN  PRINT "Falsche Eingabe! Bitte wiederholen" :  GOTO 30
70    POP Z
80    PRINT "Die eingegebene Zahl ist ",Z
```

Bild 5. Aufzählung der Funktionsunterprogramme

Ansteuerung des LCDKEY

Zur Initialisierung muß vor Gebrauch der Ausgabe-Routinen das Display initialisiert werden. Dies erfolgt mit dem Befehl „CALL 1". Dieser Befehl braucht nur einmal in einem Programm durchgeführt zu werden. Die normale Ausgabe des „PRINT" Statements, kann nach Ausführung von „UO1" auf das LC-Modul umgeleitet werden. Jederzeit kann jedoch mit „UO0" wieder auf die serielle Terminalschnittstelle zurückgeschaltet werden.

An Steuerzeichen werden folgende ASCII-Werte ausgewertet:

04H blinkender Cursor ein
05H Cursor aus
08H ein Zeichen nach links (max. bis Zeilenanfang)
0AH neue Zeile (Wechsel zwischen 1. und 2. Zeile, kein Scroll)
0DH Cursor an Zeilenanfang

Andere Steuercodes sind zur Zeit nicht implementiert und werden ignoriert (sonstige ASCII-Werte zwischen 0 und 1FH). Zur Ansteuerung der internen Register des LC-Moduls dienen die Unterprogramme „CALL 2" und „CALL 3".

Das Tastenfeld

Das Unterprogramm zur Ansteuerung des Tastenfeldes auf dem LCDKEY-Modul ist mit einer SHIFT- und CONTROL-Funktion ausgestattet. Dabei ist die Tastenposition dieser SHIFT- und CONTROL-Tasten frei programmierbar. Es kann für jede Tastenkombination (Normal, Shift, Control, Shift+Control) ein Eintrag in einer Tabelle abgelegt werden, und damit auch der jeweilige Tastencode. Die derzeitige Belegung kann *Bild 6* entnommen werden.

VALUE

Diese Routine dient der Umwandlung eines Strings in eine Fließkommazahl. Bei INPUT-Befehlen am LCDKEY wird bei Eingabe einer falschen Fließkommazahl die Meldung „TRY AGAIN" ausgegeben und dabei das Display gelöscht. Um dies zu vermeiden, kann jetzt ein String eingegeben werden und danach per Programm gewandelt werden.

Gleichzeitig wird eine Prüfung des Strings vorgenommen, wodurch Eingabefehler in der Software abgefangen werden können.

```
! A   " B   # C
N 7   O 8   P 9    shift

$ D   % E   & F
Q 4   R 5   S 6    control

( G   ) H   = I   - J
T 1   U 2   V 3   W +

? K   * L   / M
X del Y 0   Z .    CR

shift+control   shift
control         normal
```

Bild 6. Die gegenwärtige Belegung der Funktionstasten

PRINT@, LIST@ ... Routinen

Beim Anschluß eines seriellen Druckers und Ansteuerung mit LIST#, und PRINT# ist es bisher erforderlich, die Schnittstelle mit einer niedrigen Baudrate zu betreiben, da im internen Drukkerprogramm keine Überprüfung eines Busy-Signals vorgesehen war. Dies wurde durch Verwendung des PRINT@ Treibers ermöglicht. Außerdem ist es bei diesem Treiberprogramm vorgesehen, die Datenübertragungsparameter einzustellen. Dies geschieht mit Hilfe eines 8-Bit Datenbytes, das eine Codierung nach *Bild 7* besitzt.

Zur Initialisierung der Schnittstelle muß der Wert des Format-Bytes auf den Argument-Stack gelegt werden, danach erfolgt ein Aufruf mit CALL 9. Das kann auch während des Programmlaufs durchgeführt werden. Nach dem CALL 9-Aufruf werden die Argumente der PRINT@, LIST@ usw. Kommandos ar die serielle Druckerschnittstelle übergeben (*Bild 8*). Eine Überprüfung der TIMEOUT-Funktion ist aus dem Basic möglich durch Zugriff auf die interne Speicherstelle 1AH. Dieser Zustand kann mit „DBY(1AH).and.80H" geprüft werden. Ist dieser Wert<>0, liegt ein TIMEOUT-Fehler vor.

Unterprogramm-Aufruf

Der Aufruf der Unterprogramme erfolgt jeweils mit einem CALL N-Statement. N ist dabei die jeweilige Funktionsnummer (Bild 5). Eventuell notwendige Daten werden dabei auf den Argument-Stack übergeben. [TOS] ist hierbei die Zahl auf dem Argument-Stack, [TOS-1] ... [TOS-n] die darunterliegenden Zahlen.

```
Bit  7    6    5    4    3    2    1    0
Abk. TO   X    BSY  S    P1   P0   D1   D0

Bedeutung:
TO    TimeOut-Flag    Wird beim Ansprechen der Routine
                      das BUSY-Signal innerhalb von etwa 10 s nicht aktiv,
                      wird dieses Bit gesetzt und die weitere Ausgabe
                      unterdrückt, bis das Bit wieder gelöscht ist.
X                     z.Zt. nicht benutzt
BSY                   0=aktiv low Busy / 1=aktiv High Busy
S                     0=1 Stopbit / 1=2 Stopbit
P1,P0                 00=keine 01=ODD 10=EVEN 11=MARK Parität
D1,D0                 00=5 01=6 10=7 11=8 Datenbits
```

Bild 7. Die Bedeutung der Bits im Schnittstellensteuer-Byte

```
Datenformat  =  4800 Baud,8 Bit,  keine Parität,  1 Stopbit,
                Busy über  CTS Leitung (Busy=aktiv high)

          TO   X    BSY  S    P1   P0   D1   D0
FORMAT =  0    0    1    0    0    0    1    1    = 023H

Eingabe im Kommando-Modus:
>PUSH 23H
>CALL 9
>BAUD 4800
>LIST@
READY
>
```

Bild 8. Ein Listing wird mit 4800 Baud abgerufen

```
********************************** ! WICHTIG **********************************
Die Dioden und Taster sind von der Lötseite her zu bestücken.  Dioden zuerst
einlöten.
*******************************************************************************

Stückliste
DISP      SHARP 16255, HITACHI LM016L
IC1       8255, 71055 o.ä.
D         16 Dioden 1N4148 o.ä.
T         4 Siemens-Tastenstreifen (je 4 Taster)
RNW1      Netzwerk 9pol, SIL 8*10 kOhm (22 kOhm)
P1        Poti 20 kOhm bis 50 kOhm
C1        10 mikroF/10 V, Tantal
C2        100 nF Vielschicht
ST2       Pfostensteckerleiste 2reihig 40pol.

Steckerbelegung

ST1 Buchsenleiste für LC-Modul
1    GND
2    +5 V
3    KONTRAST-SPANNUNG 0-5 V
4    RS
5    R/W
6    E
7    D0
8    D1
9    D2
10   D3
11   D4
12   D5
13   D6
14   D7


ST2 Anschluß an Basic-Emuf PIO-Stecker

1                          2
3                          4
5                          6
7                          8
9    /WR                   10   /RD
11   RESET                 12   /PIOST
13   D 0                   14   GND
15   D 1                   16   A1
17   D 2                   18   A0
19   D 3                   20
21   D 4                   22
23   D 5                   24
25   D 6                   26
27   D 7                   28
29   +5 V                   30
31                         32
33                         34
35                         36
37                         38
39                         40
```

Bild 9. Das ist die Stückliste

Rolf-Dieter Klein

NDR- und mc-CP/M-Computer: IBM-kompatibel

Der Industriestandard-Computer hat sich durchgesetzt. Wobei die Bezeichnung „Industriestandard" eine bemüht neutrale Umschreibung der Tatsache ist, daß es sich um den von IBM entwickelten PC handelt. Der Softwaremarkt bietet heute eine solche Fülle von Programmen für den IBM-Computer, der Hardwaremarkt eine solche Fülle von Karten, daß es für alle anderen Computerhersteller Pflicht ist, sich einen Weg in die IBM-Welt zu bahnen. Also: auch der NDR-Computer wird kompatibel.

Hier wird eine Buskopplung vorgestellt, die beide Welten so verbindet, daß ein System daraus entsteht, in dem man Original-IBM-Karten verwenden kann. Man kann sogar im System eine NDR- und eine IBM-CPU parallel arbeiten lassen. Der IBM-Teil kann in den NDR-Teil hineinsehen – ohne die Arbeit des NDR-Computers zu stören. Die Baugruppe besitzt zwei Stecker für die mc-NDR-Welt, einen für den NDR-Computer und einen VG-Stecker für den Einsatz im mc-CP/M-Computer. In einer solchen Koppelung müssen die CPU-Baugruppen DMA-fähig sein, sonst funktionieren sie nicht.

Was heißt hier kompatibel?

Welche Bedingungen muß ein Computer erfüllen um IBM-kompatibel zu sein? Zunächst einmal braucht man eine der CPUs, die im IBM verwendet werden, also entweder einen 8088 oder 80286 oder gar einen 80386. Der IBM-Computer arbeitet mit MS-DOS. Dieses Betriebsystem kann man im Prinzip so anpassen, wie man es vom alten CP/M her kennt. Das MS-DOS besitzt eine sauber definierte Schnittstelle, die mit sogenannten Software-Interrupts arbeitet. Damit hätte man ein portables Konzept und man wäre unabhängig von Hardwaredetails, da alle Schnittstellen, wie Floppy, Bildschirm und Tastatur, durch Softwareschnittstellen standartisiert ansprechbar sind. Leider haben sich die Softwareproduzenten beim Programmieren oft nicht daran gehalten. So sind Programme, die zum Beispiel direkt auf den Bildwiederholspeicher des IBM zugreifen, eher der Normalfall. Aber auch der Floppy-Controller wird direkt angesprochen, um zum Beispiel einen Kopierschutz zu bewerkstelligen. Der Tastaturtreiber des IBM-Computers produziert einen Code, der nicht mit ASCII beziehungsweise mit ISO übereinstimmt. So ist man gezwungen, will man IBM-kompatibel sein, wirklich die Hardware des IBM-Computers weitestgehend nachzubauen, anfangen von der Tastatur über den Floppy-Controller, die Bildschirmschaltung bis hin zu innersten Details.

Der IBM-Computer ist darüberhinaus in seinem Aufbau nicht modular konstruiert. Daher ist es kaum möglich, die Modularität des NDR-Systems in eine kompatible Version mit hinüber zu nehmen. Nun wollten wir aber nicht einfach einen IBM-Computer (mit seiner großen Grundplatte) neben einen NDR-Computer stellen. Wir sind daher einen anderen Weg gegangen. Seit kurzem gibt es den IBM-Computer auf einer steckbaren Platine. Dies ist durch Integration möglich geworden. Man benötigt kein großes Motherboard mehr, wenn man einen IBM-Computer aufbauen will. Es genügt eine IBM-Busplatte mit aufgesteckter CPU- sowie Floppy- und Bildschirmkarte. Fertig ist der IBM.

Mit dem NDR-Computer hat das noch nichts zu tun, der IBM-Teil ist zunächst erst einmal halbwegs modular geworden. Wenn man dabei einen AT-Bus verwendet, so kann man später auch durch Wechsel der CPU zum IBM-AT aufrüsten.

Jede der bisherigen IBM-Computerklassen kann über eine Schaltung, die die Busse beider Systeme verkoppelt, mit dem mc-NDR-System verbunden werden. Der IBM-Computer kann dabei in den Speicher des NDR-Computers hineinschauen. Und er kann auf alle IO-Ports des NDR-Computers zugreifen. Auf dem NDR-Computer-Bus darf (muß aber nicht) auch eine CPU sitzen, die dann ihre eigenen Aufgaben erledigt.

Viele verschiedene Gesamtkonfigurationen sind möglich.

Bild 1 zeigt die einfachste aber umfangsreichste Variante. Beide Systeme sind komplett eigenständig. So gibt es zwei Tasturen und zwei Bildschirmkontroller. Dabei ist nur ein Bildschirm notwendig, denn auf der Koppelplatine ist ein Bildschirmumschalter vorgesehen. Weil ein IBM-Kompatibler mit 40-Spur-Laufwerken und der NDR normalerweise mit 80-Spur-Laufwerken arbeitet, sind zwei Floppy-Laufwerke nötig.

Bild 2 zeigt eine Variante, in der nur auf der IBM-Seite Peripherieeinheiten angeschlossen sind. Der NDR-Computer greift auf diese Einheiten mit Hilfe be-

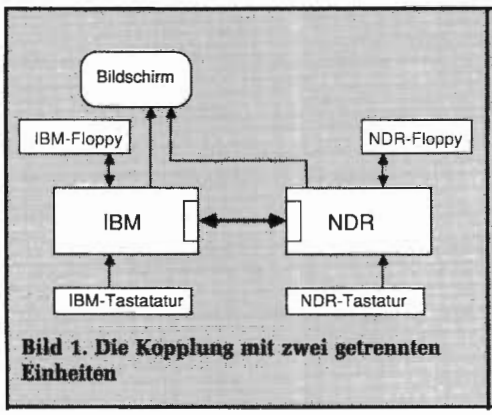


Bild 1. Die Kopplung mit zwei getrennten Einheiten

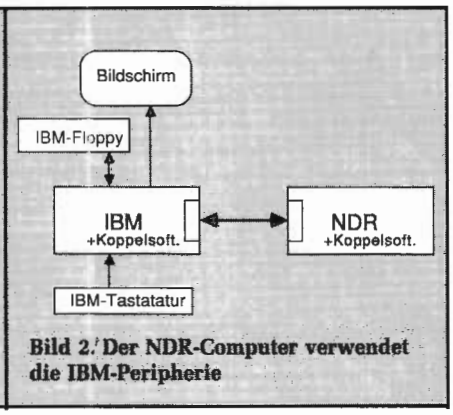


Bild 2. Der NDR-Computer verwendet die IBM-Peripherie

sonderer Koppelsoftware zu. So legt die NDR-CPU in einer bestimmten Speicherzelle einen Zugriffswunsch ab, und ein auf dem IBM laufendes Programm fragt diese Zelle ständig ab. So kann man zum Beispiel Bildschirmausgaben auf den IBM umleiten aber auch Tastatureingaben vom IBM auswerten und sogar das Floppy-Laufwerk (dann aber nur mit 40 Spuren) mitverwenden.

Auf diese Art läßt sich das FLOMON-Programm beim Z80 leicht anpassen. Wenn man beim NDR-Computer nicht auf die schnelle Grafik verzichten will, so kann man die Grafikkarte des NDR-Computers verwenden und nur die Tastatur und die Floppy des IBM-Gerätes nutzen.

Die verwendete NDR-CPU muß wie gesagt DMA-fähig sein, um mit dieser Kopplungsart arbeiten zu können. Derzeit gilt das nur für die Z80-CPU (Vollausbau, oder SBC3), die 64180-CPU und die 68020-CPU (32-Bit Rechner). Nicht jedoch für 68008 und 68000, die man jedoch leicht umrüsten kann., wie noch in einem späteren Artikel beschrieben wird.

Aufbau und Wirkungsweise der Schaltung

Bild 3 zeigt eine Blockschaltbild der gesamten Anordnung. Links ist die IBM-Koppellogik zu sehen und rechts der Schaltungsteil des NDR-Computers. Beide Teile sind über ein 50-poliges Flachbandkabel miteinander verbunden.
Im IBM-Adreßraum belegt der NDR-Computer nur 64 KByte für Speicher und 32 KByte für IO. Um bis zu 1 MByte im NDR-Computer erreichen zu können, wird ein Latch genutzt, das „Bänke" umschaltet. Dorthinein wird die zusätzliche Adreßinformation gespeichert.

Bild 4 zeigt den Schaltungsteil für die IBM-Seite. Die Adreßleitungen A0 bis A15 werden dabei mit Hilfe zweier Bustreiber (74LS244) ständig durchgeschaltet. Die Adreßleitungen A16 bis A19 kommen von dem Latch IC 74LS273. Dieses Latch belegt acht IO-Adressen im IBM-Adreßraum und zwar von 300h bis 307h, da diese nicht eindeutig dekodiert sind. Dieser Adreßraum ist im IBM für Prototypenadapter reserviert. Wenn man auf eine dieser Adressen schreibt, so wird der Wert am Datenbus in das Latch gespeichert. Die IO-Adresse auf der IBM-Seite kann man in der Schaltung durch Ändern der Brücken am IC 74LS688 auch anders

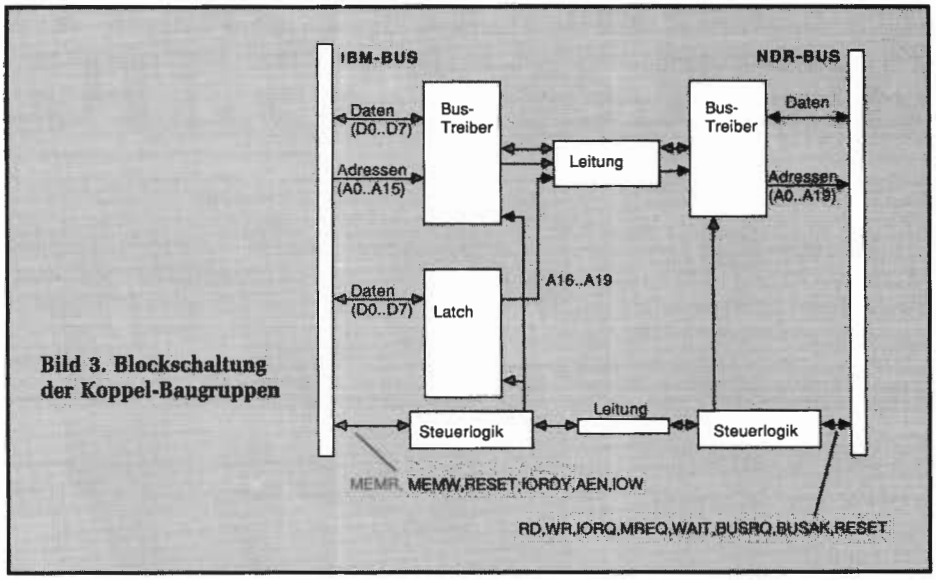


Bild 3. Blockschaltung der Koppel-Baugruppen

einstellen, im Layout sind entsprechende Brücken vorgesehen.

Das IC 74LS245 ist ein bidirektionaler Bustreiber, der D0 bis D7 auf die Leitung schaltet. Dabei schaltet der Bustreiber die Datenrichtung nur dann zum IBM-Bus, wenn der IBM lesend auf einen der gültigen Bereiche zugreift. Die Bereiche werden durch einen PAL-Baustein bestimmt, deren Gleichungen ebenfalls im Bild sichtbar sind.
Zur Verknüpfung werden die Adreßleitungen A14 bis A19, sowie die Steuerleitungen MEMR, MEMW, AEN und RESET DRV vom IBM an das PAL geführt. Es erzeugt damit auch die Signale IORQ, MREQ, RD, WR und RESET, sowie das DIR-Signal. Ein Zugriff auf den Bereich DC000 bis DFFFF wird als IO-Zugriff in den NDR-Computer ausgewertet. Daher erscheint dann das Signal IORQ. Bei einem Lesezugriff werden ferner das Signal RD und das Signal DR auf 0 gelegt, bei einem Schreibzugriff nur das Signal WR. Bei einem Zugriff auf den Bereich E0000 bis EFFFF wird das Signal MREQ anstelle von IORQ aktiviert. Damit gibt es einen Speicherzugriff in den NDR-Computer.
Das Signal RESET DRV ist beim IBM-Computer High-Aktiv und wird daher durch das PAL invertiert, um so z.B. das Latch 74LS273 nach dem Einschalten rückzusetzen. Ferner ist es dann mit dem NDR-Bus verträglich. Das Signal WAIT wird von der Leitung über einen Bustreiber an den IBM-Bus auf das Signal I/O-RDY geschaltet. Der Bustreiber ist dabei so verschaltet, daß er nur dann aktiv ist, wenn ein 0-Signal an seinem Eingang liegt. Damit wird ein offener Kollektor nachgebildet. Das Latch liefert noch vier weitere Signale, die mit 'relais' und 'a1','b1' und 'c1' beschriftet sind. Sie dienen zu späteren Erweiterungen und werden für die Grundschaltung zunächst nicht benötigt.
Die Signale, die auf die Verbindungsleitung zum NDR-Computer führen, entsprechen schon direkt denen des NDR-Busses. Man könnte sie direkt mit dem Bus verbinden. Dann wäre es aber nicht möglich, auf dem NDR-Bus auch gleichzeitig eine eigene CPU zu betreiben. Ferner ist die Leitungslänge bei direkter Verbindung kritisch.

Bild 5 zeigt die Schaltung auf der NDR-Bus-Seite. Die Steuerschaltung besteht auch hier wieder aus einem PAL-Baustein. Daten und Adressen werden auf den NDR-Bus genau dann durchgeschaltet, wenn das Signal 'sel1' auf 0 liegt. Die Steuerleitungen, RD, WR, IORQ und MREQ werden mit 'sel2' freigegeben.

Zwei Betriebsarten.

Befindet sich keine CPU auf dem NDR-Bus, dann muß man die Brücke J1 am PAL schließen und damit werden 'sel1' und 'sel2' fest auf 0V gelegt. Der IBM hat dann dauerhafte Verbindung mit dem NDR-Bus.
Ist eine CPU vorhanden, so muß die Brücke offenbleiben und dann geschieht folgendes: Wenn der IBM einen Zugriff auf den NDR-Bus durchführen will, so legt er entweder MREQ oder IORQ auf die Leitung. Das PAL schaltet diese Signale in Form eines BUSRQ (Bus Request) auf den NDR-Bus. Das PAL sorgt ferner auch dafür, daß der IBM nun zunächst einmal wartet, indem es das Signal WAIT auf 0 V legt. *Bild 6* zeigt den

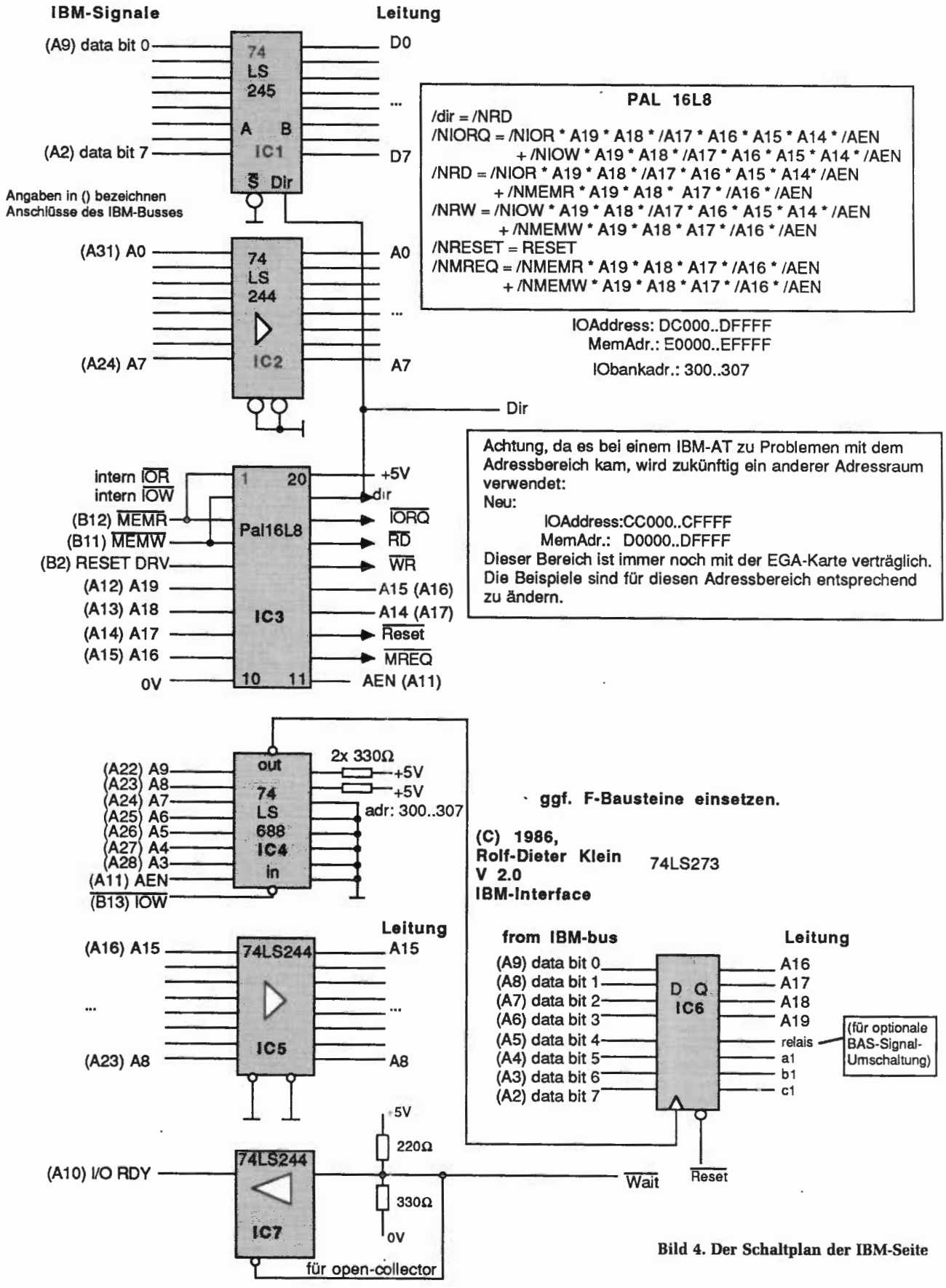
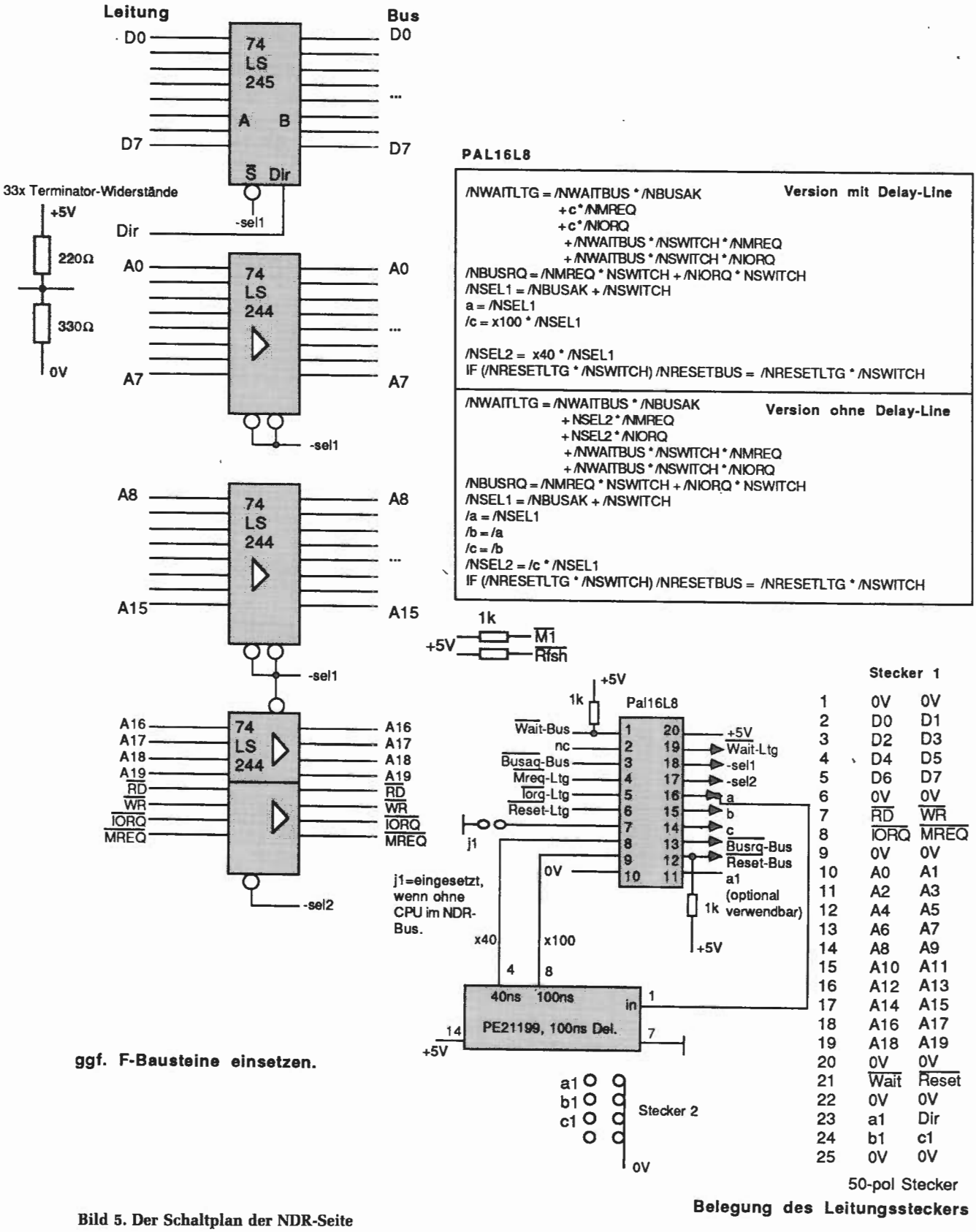


Bild 4. Der Schaltplan der IBM-Seite



ggf. F-Bausteine einsetzen.

Belegung des Leitungssteckers

Bild 5. Der Schaltplan der NDR-Seite

kompletten Zeitablauf. Nun bestätigt die CPU auf dem NDR-Bus nach einiger Zeit mit dem Signal BUSAK, daß der NDR-Bus frei ist.
Jetzt wird vom PAL das Signal 'sel1' auf 0 geschaltet und damit werden zunächst Daten- und Adreßbus auf den NDR-Bus geschaltet. Dies muß zuerst geschehen, sonst gibt es Probleme bei der Adreßerkennung. Nach ca. 50ns wird auch das Signal 'sel2' auf 0V gelegt. Damit werden die Signale IORQ, MREQ, RD und WR gültig. Bis zu dieser Zeit wurde das WAIT-Signal von dem PAL auf 0 gelegt und die IBM-CPU gebremst. Nun wird das WAIT-Signal ggf. von der NDR-Baugruppen bedient. Eine kleine Verzögerung tritt nun dadurch auf, daß die NDR-Baugruppen natürlich auch eine Weile brauchen, um zu erkennen, daß sie adressiert sind. Dadurch entsteht ein kleiner Puls auf der Wait-Leitung. Er ist ca. 40 ns breit. Er stört solange nicht, als er schmäler als die halbe Periodendauer des Betriebstaktes der IBM-CPU ist, da dort das WAIT-Signal (bzw. I/O-RDY) mit den Taktflanken des Prozessortaktes synchronisiert wird.
Will man den Puls vermeiden, so muß man die in die Schaltung eingezeichnete Verzögerungsleitung verwenden und das PAL mit anderen Gleichungen (ebenfalls eingezeichnet) belegen. Für die vorgeschlagene CPU mit dem 8088 reicht aber die einfache Version ohne Verzögerungsleitung aus. Die Leitung 'a1' ist ebenfalls auf den PAL-Baustein gelegt, um ggf. spätere Erweiterungen zu ermöglichen.

Bild 7 zeigt die Schaltung des BAS-Mischers und der Umschaltlogik. Der BAS-Mischer ermöglicht es, einen normalen Monitor als Ausgabegerät für den IBM-Teil zu verwenden. Der Umschalter erlaubt es wahlweise, den IBM- oder den NDR-Computer-Bildschirmausgang an den Monitor zu schalten. Dabei gibt es zwei Möglichkeiten. Einmal kann man dies mit Hilfe des Schalters S1 tun, oder der Computer erledigt das über die Leitung 'relais', die vom 74LS273 kommt. Dabei sind die beiden Signale Exclusiv-Oder verknüpft, so daß man wie bei einer Wechselschaltung mit beiden Signalen eine Umschaltung erreichen kann.

Bild 8 zeigt die beiden Baugruppen. Die Flachbandleitung zur Verbindung ist gemäß Stecker 1 wie in *Bild 5* belegt. *Bild 9* zeigt den Einbau in das NDR-Gehäuse. Das Gehäuse wurde dabei schon vor einiger Zeit für die IBM-Slots vorbereitet und auch die meisten NDR-Baugruppen wurden auf ein neues Format gebracht,

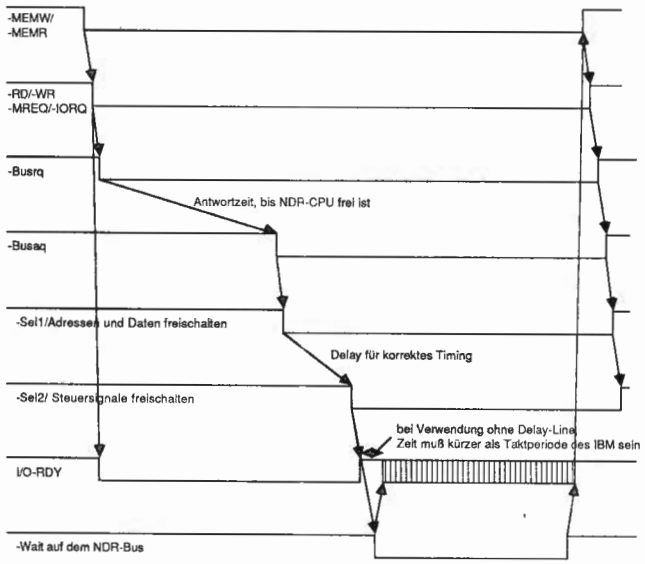


Bild 6. Zeitlicher Ablauf bei einem Zugriff

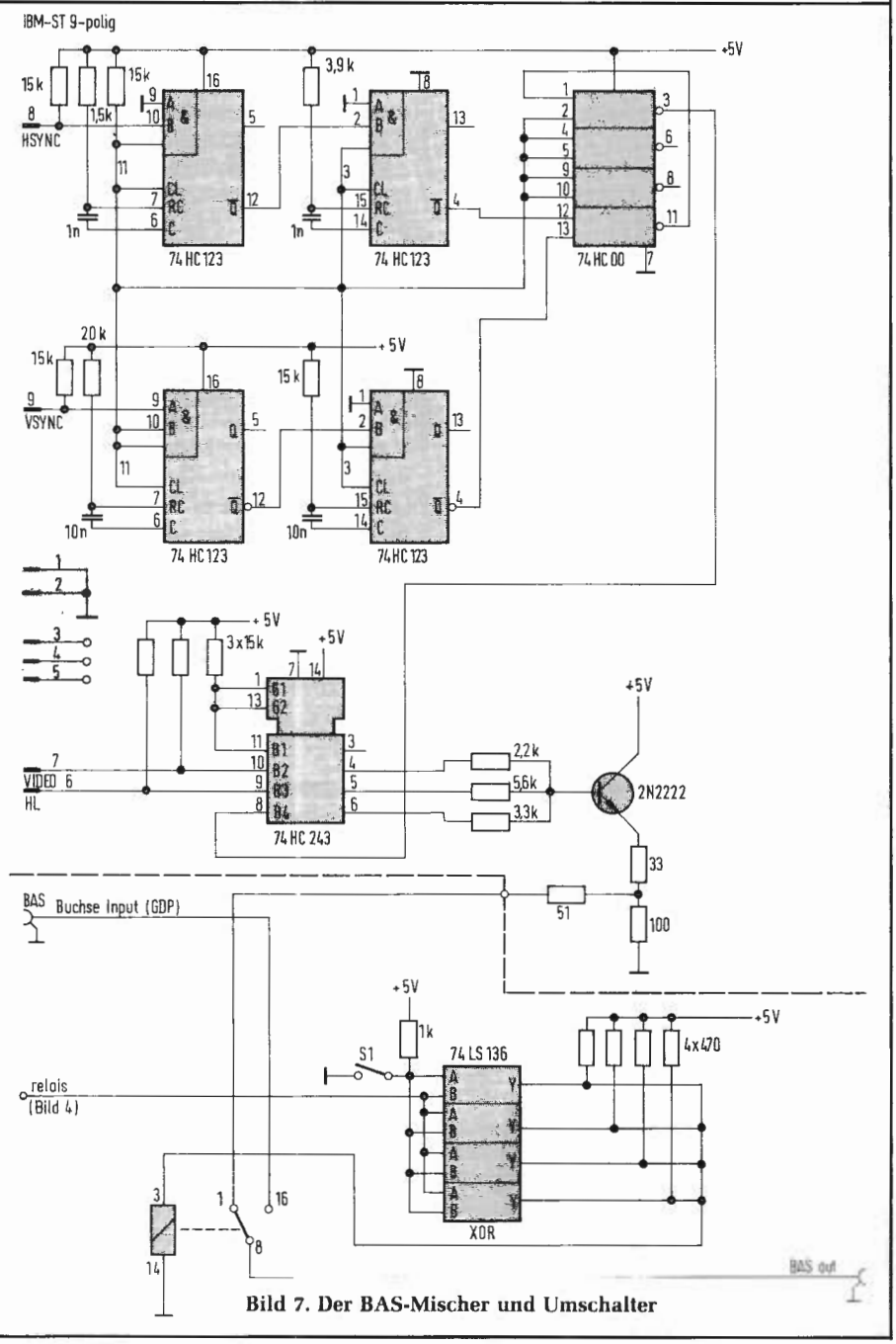


Bild 7. Der BAS-Mischer und Umschalter

so daß alle Verbindungen einheitlich an der Gehäuseseite herausgeführt sind.

Selbst bauen?

Für alle diejenigen, die den Selbstbau wagen wollen, zeigt *Bild 10* die Belegung des IBM-Busses. Wenn man nicht die fertige Leiterplatte verwenden will, so kommt man am ehesten mit einem IBM-Prototypen-Adapter zurecht. Dieser ist allerdings sicher teurer als der ganze Adapter-Bausatz.

Programmbeispiele für die Kopplung

Bild 11 zeigt eine Übersicht des IBM-Adreßraums und der Zuordung zum NDR-Adreßraum. Der IO-Adreßraum belegt vom IBM aus gesehen 32 KByte, der NDR-Computer wertet derzeit aber nur die unteren 8 Bit aus, also 256 verschiedene Adressen. Die IOs liegen im IBM von DC000 bis DFFFF. Dabei ist der Bereich DC000 bis DC0FF zum Beispiel ein sinnvoller Adreßbereich. Beim Speicher ist das Problem eher umgekehrt. Im IBM wird der Adreßraum von E0000 bis EFFFF, also 64 KByte belegt, im NDR muß aber der Bereich 00000 bis FFFFF angesprochen werden können. Dazu werden die unteren Adressen direkt vom IBM übernommen, also 0000 bis FFFF und die oberen vier Bit von einem Latch hinzugefügt. Das Latch ist auf dem IBM im IO-Adreßraum bei Adresse 300..307 ansprechbar. Will man auf eine beliebige Speicherzelle im NDR-Computer zugreifen, so muß man zunächst das Latch belegen und kann danach im Bereich E0000 bis EFFFF auf den entsprechenden Bereich zugreifen. Wenn man die Bank-Boot-Baugruppe verwendet oder die SBC3, so kann das derzeit zu einem Buskonflikt führen, da in dieser Baugruppe bisher keine Tristate-Treiber

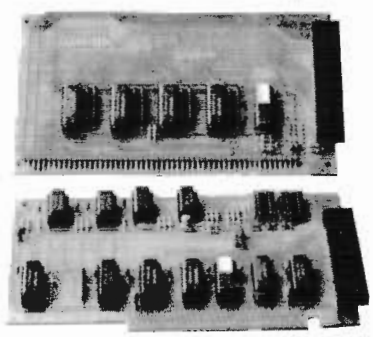


Bild 8. Diese beiden Platinen verbinden Welten

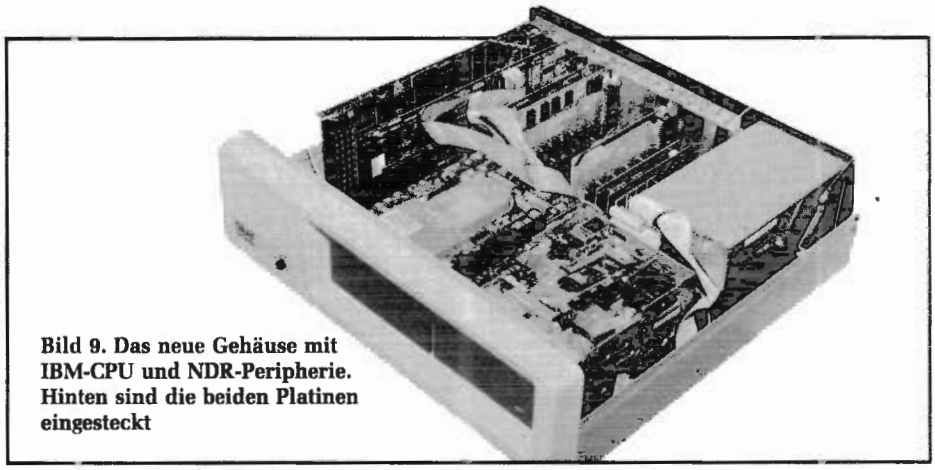


Bild 9. Das neue Gehäuse mit IBM-CPU und NDR-Peripherie. Hinten sind die beiden Platinen eingesteckt

für die Adreßleitungen A16..A19 verwendet wurden. Im normalen CP/M-Betrieb jedoch passiert nichts, da hier nur der Adreßbereich 00000 bis 0FFFF verwendet wird. In das Latch wird man dabei immer den Wert 0 schreiben. Ein anderes kleines Problem gibt es zusammen mit der Bank/Boot-Logik. Wenn der Z80 durch einen Zugriff auf Port C8 den unteren Speicherbereich x0000 bis x7FFF ausblendet und durch den Inhalt der Boot-EPROMs ersetzt, so kann auch der IBM nicht auf diesen Bereich zugreifen. Dies müßte man durch Semaphor-Technik später einmal berücksichtigen (FLOMON müßte angepaßt werden). Die Umschaltung geschieht bei FLOMON immer dann, wenn der Z80 Dienstprogramme (wie Consol-Ausgabe) aufruft. So ist es für ein universelles Kommunikationsprogramm zunächst einfacher im oberen Bereich zuzugreifen, also von 08000 bis 0FFFF.

Leuchtdioden an und aus:

Ein einfaches Beispielprogramm zeigt *Bild 12*. Das Programm ist in Turbo-Pascal geschrieben, und läßt sich daher leicht in andere Sprachen übertragen. Das Programm hat die Aufgabe, acht Leuchtdioden binär durchzuzählen. Die Leuchtdioden befinden sich auf der IOE-Baugruppe des NDR-Computers, an Port 30h. Die Prozedur LEDSAN gibt einen beliebigen Datenwert an dieses Port aus. Dazu wird mit dem MEM-Befehl von Turbo-Pascal gearbeitet. In Klammern stehen bei MEM zwei Angaben. Der erste Wert ist der sogenannte Segment-Offset. Er ist bei uns für den IO-Bereich fest und liegt bei DC00h. Der Segementoffset wird bei der Adressierung durch den 8088 zu einem Adreßwert addiert, nachdem er um vier Bit nach links geschoben wurde. Dieser Adreßwert ist bei uns 30h, denn dort liegt die IOE-Baugruppe im NDR-Adreßraum. Durch eine Zuweisung erhält dieses Port den Wert der Variablen i, die als Parameter bei Aufruf der Prozedur LEDSAN übergeben wurde.

Im Hauptprogramm wird nun eine weitere Variable i von 0 bis 255 durchgezählt und in diesem Block auch LEDSAN mit i als Parameter aufgerufen. Der Aufruf von DELAY(50) sorgt außerdem dafür, daß nach jedem Hochzählen ein klein wenig gewartet wird, so daß das Auge den Zählvorgang an den LEDs verfolgen kann. Die Zählschleife ist schließlich noch durch REPEAT UNTIL geklammert und wird solange durchlaufen, bis man irgendeine Taste (KEYPRESSED) drückt.

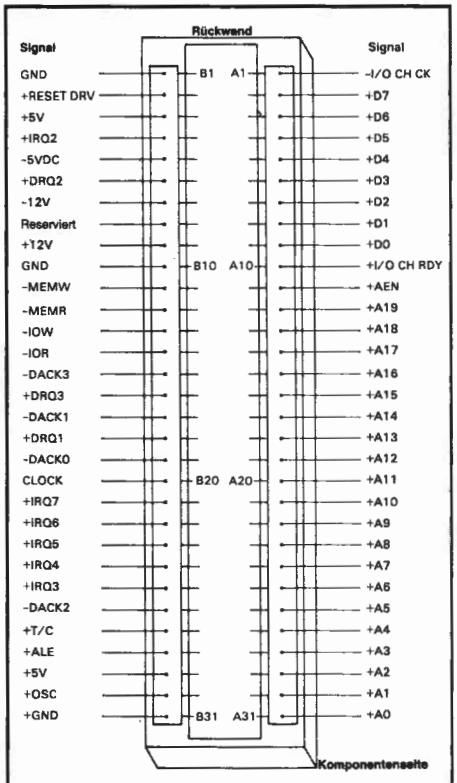


Bild 10. Die Belegung des IBM-Busses

Schalter einlesen

So einfach wie mit dem Schreiben verhält es sich auch mit dem Lesen von Daten. *Bild 13* zeigt ein Beispielprogramm. Hier wird von der IOE-Baugruppe gelesen.

Wenn man dort zum Beispiel eine Schalterreihe mit den Eingängen von Port 30h verbindet, so kann man den Schalterzustand direkt am Bildschirm in binärer Schreibweise sehen. Dabei ist Bit 0 links auf dem Bildschirm angeordnet. Die Schalterstellungen werden durch das Programm solange ausgegeben bis man eine Taste drückt.

Speicher testen

Bild 14 zeigt ein kleines Programm, das bei der Inbetriebnahme der Baugruppen ganz nützlich sein kann. Es setzt zunächst Port 300 auf 0, und wählt damit den Adreßraum 00000 bis 0FFFF im NDR-Computer aus. Dann wird der Inhalt der Speicherzelle 00000 gelesen und gleich wieder geschrieben. Nun kann man z.B. mit einem Oszilloskop den Zugriff und die Wait-Signal-Erzeugung kontrollieren. Wenn man den Test mit eingesteckter NDR-CPU fährt, sollte man allerdings besser eine der Adressen im Bereich 08000 bis 0FFFF verwenden, indem man bei MEM nach dem Doppelpunkt z.B. $8000 angibt.

Speicher anzeigen

Interessant ist es, den Speicherinhalt auf den Bildschirm des IBM zu bringen. Dazu dient das Programm in *Bild 15*. Es gibt den Speicherbereich 0F000 bis 0FFFF in sedezimaler Schreibweise auf dem IBM-Bildschirm aus. Andere Bereiche kann man leicht durch Änderung der FOR-Schleife ausgeben. Wer will, kann das Programm zu einem richtigen kleinen Monitorprogramm ausbauen, mit Funktionen wie Speicher-Lesen, Speicher-Ändern usw.

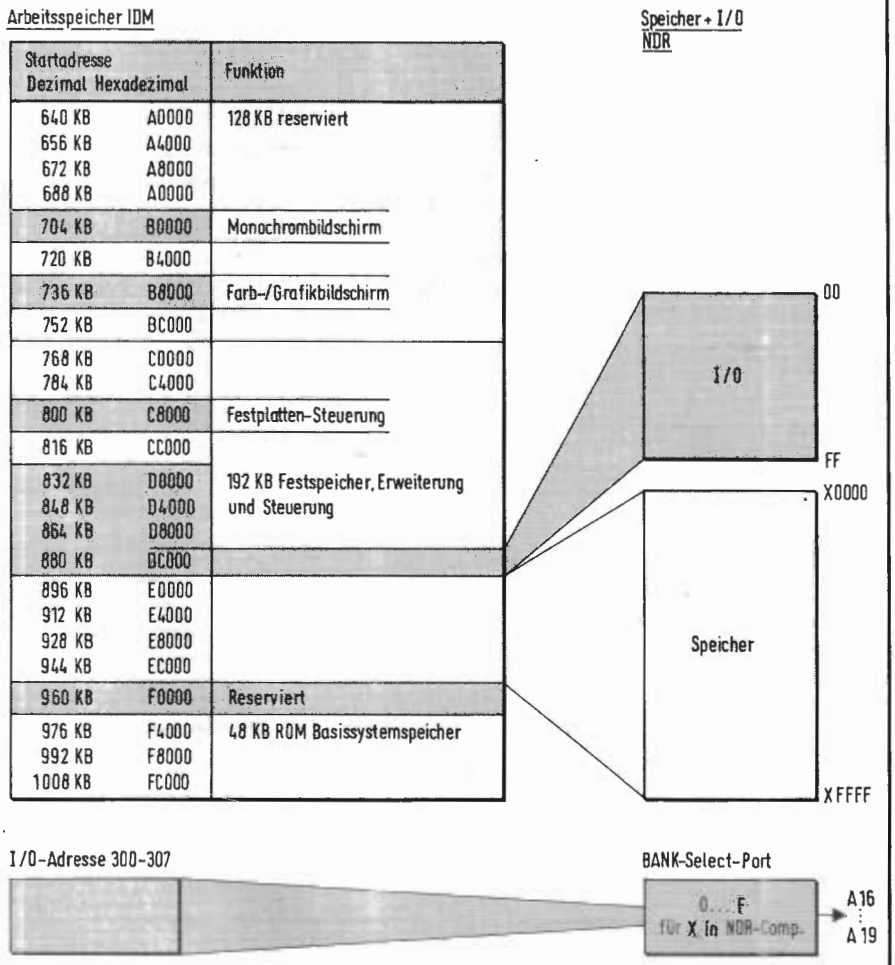


Bild 11. Die Adreßraumzuordung

Stoppt den Z80

Manchmal stört es, wenn die Z80-CPU arbeitet. In *Bild 16* ist dargestellt, wie man sie anhalten kann. Man überschreibt den Speicherbereich, in dem die CPU gerade arbeitet, mit Halt-Befehlen. Das Programm arbeite z.B. mit FLOMON. Der Z80 muß dann ständig im Bereich 0F000 bis 0FFFF arbeiten. Er springt von dort aus in die Bank/Boot-EPROMs und kehrt dann wieder zurück. Wenn man den Bereich 0F000 bis 0FFFF daher mit Halt-Befehlen überschreibt, muß der Z80 irgendwann stehen bleiben. Anschließend kann man sich den Bereich einmal mit dem Programm aus Bild 15 ansehen. Man wird feststellen, daß unter Umständen nicht alle Zellen der Wert 76h (Halt-Befehlscode) enthalten, da der Z80 auch seinen Stack im Bereich vor FFFF betreibt, und er vor Erreichen eines Halt-Befehls dort noch Werte ablegen kann. Wenn der Z80 einmal steht, kann man zum Beispiel auch mit der GDP direkt vom IBM aus arbeiten, sonst wird sie vom FLOMON verwendet.

GDP löschen

Bild 17 zeigt ein einfaches Programm, das einen Lösch-Befehl an die GDP-Baugruppe schickt. Sicherheitshalber sollte man allerdings auch Port 60h vor Einsatz des Befehls auf 0 stellen, sonst wird nur die aktuelle Bildseite gelöscht.

GDP Textausgabe

Wie man einen Text auf den Bildschirm der GDP bringt zeigt *Bild 18*. Hierzu sollte man den Z80 vorher lahmlegen. Doch probieren Sie es auch einmal mit laufendem FLOMON aus, es können jedoch dabei Störungen auftreten, wenn der Z80 zum Beispiel die Bildseite gerade umschaltet, bevor der IBM seine Ausgabe gemacht hat.

Der Z80 als intelligentes Terminal

Eine interesante Anwendung ergibt sich bei FLOMON: echter Multiprozessor-Betrieb. Der Z80 dient als intelligentes Terminal mit Grafik-Ausgabe. Der IBM kann dann dort Bilder zeichnen. Die Anordnung ähnelt dem Verhalten der TERM1. Dazu wird ein Programm auf dem NDR gestartet, das wie folgt aussieht:

```
START:
XOR A
LD (8000h),A
;löschen der Kommunikationszelle
```

```
program ledstest;

 var i:integer;

procedure ledsan(i:integer);
begin
 mem[$dc00:$30] := i;
end;


begin
 writeln('Rolf-Dieter Klein, IBM-NDR Kopplung');
 writeln('LEDs auf Adresse 30h, IOE werden angesteuert');
 repeat
  for i:=0 to 255 do begin
   delay(50);
   ledsan(i);
  end;
 until keypressed;
end.
```

Bild 12. Leuchtdioden auf der IOE schalten

```
program switchtest;


procedure schalteraus;
 var val,i : integer;
begin
 val := mem[$dc00:$30];
 for i:=0 to 7 do begin
  write(val mod 2);
  val := val div 2;
 end;
 writeln;
end;

begin
 writeln('IBM - NDR, Rolf-Dieter Klein');
 writeln('Schalter-Test Port 30h');
 repeat
  schalteraus;
 until keypressed;
end.
```

Bild 13. Schalter auf der IOE ablesen

```
program memtest;

var i : integer;

begin
 port[$300] := 0; (* Bank 0 anwaehlen *)
 repeat
  i:=mem[$e000:0];
  mem[$e000:0] := i;
 until keypressed;

end.
```

Bild 14. Testprogramm für den Speicherzugriff

```
program spdump;

var i : integer;

procedure hexaus( n : integer);
procedure nibble( i:integer);
begin
 if (i >= 10) then
  write(chr(i -10 + ord('A')))
 else
  write(chr(i + ord('0')));
end;

begin
 nibble(n div 16);
 nibble(n mod 16);
end;


begin
 port[$300] := 0;
 for i:=$f000 to $ffff do begin
   if (i mod 16) = 0 then writeln;
   hexaus(mem[$e000:i]);
   write(' ');
 end;
end.
```

Bild 15. Programm zum Speicherdump

```
program haltz80;
var i : integer;
begin
 port[$300] := 0;
 for i := $f000 to $ffff do begin
  mem[$e000:i] := $76;
 end;
end.
```

Bild 16. Der Z80 wird angehalten

```
program clrgdp;

begin


 mem[$dc00:$70]:= 7;
end.
```

Bild 17. Befehl an die GDP-Baugruppe

```
program gdptest;
var i,x,y: integer;
 str: string[100];

procedure befgdp(i : integer);
var k:integer;
begin
 repeat
  k := mem[$dc00:$70];
 until (k and 4) <> 0;
 mem[$dc00:$70] := i;
end;

procedure waitgdp;
var k:integer;
begin
 repeat
  k := mem[$dc00:$70];
 until (k and 4) <> 0;
end;

begin
 writeln('Achtung, vorher mit halt.pas z80 stoppen');
 mem[$dc00:$60] := 0; (* Seite 0 *)
 befgdp($4);
 befgdp($d);
 befgdp($e);
 waitgdp;
 mem[$dc00:$73] := $33;
 str := 'Daten vom IBM an den NDR';
 for i:=1 to length(str) do
  befgdp(ord(str[i]));
waitgdp;
mem[$dc00:$73] := $11;
end.
```

Bild 18. Kleines GDP-Testprogramm

```
program exec;

var ch : char;

procedure cmdout(i : integer);
var n : integer;
begin
 repeat
  n := mem[$e000:$8000];
 until n=0;
 mem[$e000:$8000] := i;
end;

begin
 port[$300] := 0;
 cmdout($1a);
 repeat
  read(kbd,ch);
  cmdout(ord(ch));

 until ord(ch) = 1;
 cmdout(2);
end.
```

Bild 19. Der NDR-Computer als TERM1

```
SCHLEIFE:
LD A,(8000h)
;warten bis Befehl da
OR A
;0= kein Befehl da
JR Z,SCHLEIFE
LD C,A
;Befehl ausführen
CALL 0F009h
;CO-Routine im FLOMON
JR START
;Befehl löschen und dann warten
```

Bild 19 zeigt den Programmteil auf dem IBM. Hier als Test programmiert, kann man nun auf der IBM-Tastatur Text oder Befehle eintippen und erhält eine Ausgabe auf dem NDR.
Wenn man z.B. die Sequenz:

ESC ESC G CR
M0 0 CR
R100 100 CR

eingibt, erhält man ein Rechteck auf dem Bildschirm. Die verwendeten Grafik-Befehle sind übrigends im Sonderheft Mikrocomputer Schritt für Schritt 2 beschrieben, bzw. im Buch „Rechner modular", von R. D. Klein, Franzis-Verlag.

Sven B. Schreiber, Josef Bernhardt

Digitalisieren mit „Kompatiblen“

Der in Heft 10/85 vorgestellte Video-Digitalisierer [1] war eine gefragte Sache. Viele haben bedauert, daß bisher nicht beschrieben worden ist, wie man ihn an einen IBM-PC-kompatiblen Computer mit Standard-Grafik (CGA-Karte) anschließt. Wir holen das hiermit nach.

Weil man einen Parallel-Ein-/Ausgabe-Baustein ohnehin für viele Zwecke gut gebrauchen kann, benutzen wir den PPI 8255 als Interface zum Digitalisierer. Welche Bussignale verwendet der Digitalisierer? Zunächst einmal einen bidirektionalen 8-Bit-Datenbus, dann einen 20-Bit-Adreßbus (Eingänge) und schließlich vier Steuerleitungen namens S0...S3 (ebenfalls Eingänge), die den Datenaustausch mit dem Prozessor regeln.
Das Rechneranpassungsfeld des Digitalisierers muß zunächst richtig verdrahtet sein. Um Ihnen die Arbeit zu erleichtern, geben wir hier gleich die zu verbindenden Anschlüsse an: 1–20, 3–18, 8–13, 14–16 (*Bild 1*). Die Signale S0, S1 und S2 entsprechen dann den Bussignalen Read (RD), Write (WR) und Nicht-I/O-Request (/IORQ). Daß RD und WR aktiv-high definiert sind, hat, wie Sie später sehen werden, seine ganz besondere Bewandtnis.
Da der Rechnerbus nicht in allen Details nachgebildet werden muß, können einige Signale entfallen, beispielsweise die Adreßleitungen A3...A19. Sie dürfen wahlweise auf Null oder Eins liegen, je nachdem, wie die Schalter 1 und 2 eingestellt sind. Die Brücke für A19, die zum 74LS688/1 führt, muß so eingestellt sein, daß Pin 1 (/G) auf jeden Fall auf Null liegt. Wenn alles stimmt, sollte Pin 19 (/P=Q) des 74LS688/2 permanent ein Low-Signal liefern. Dadurch werden die beiden Adreßdecoder 74LS138/1 und 74LS138/2 „scharf“ gemacht.

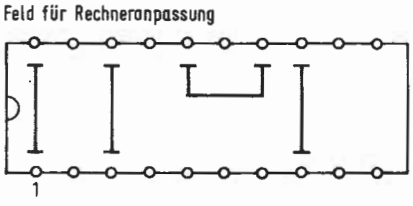


Bild 1. So ist das Rechneranpassungsfeld des Digitalisierers mit Brücken versehen

Als absolut lebensnotwendige Bussignale bleiben also D0...D7, A0...A2 und S0...S2. Wie aus der *Tabelle* zu ersehen, wurde der 8-Bit-Datenbus dem PPI-Port A zugewiesen, die restlichen Signale, allesamt Eingänge, dem Port B. Da der Datenbus des Digitalisierers bidirektional betrieben wird, hat Port A ständig nach Bedarf zwischen Input- und Output-Modus zu wechseln. Daß dabei die Port-B-Ausgänge immer wieder auf Null gesetzt werden (siehe 8255-Datenblatt) macht nichts, denn der Digitalisierer erkennt ja nur I/O-Zugriffe an, wenn /IORQ und RD oder WR aktiv sind. Die beiden letzteren sind jedoch aktiv-high.
Eine Besonderheit des Digitalisierers verlangte die Durchführung eines kleinen Hardware-Eingriffs. Wie in [1] zu lesen, wird Bit 0 des Statusregisters, welches das Vorhandensein von Bilddaten signalisiert (READY), immer bei Schreibzugriffen auf das Steuerregister 74LS373 rückgesetzt. Darum muß man nach jeder Datenübernahme erneut ein Steuerwort auf den Bus legen, obwohl dies oftmals überhaupt nicht notwendig wäre. Ein Prozessor macht das in Nullkommanichts, der PPI hingegen verlangt dazu ein aufwendiges Hin- und Herschalten, bis diese Prozedur endlich über die Bühne ist. Da eine Videozeile ja nur 64 µs dauert, während derer auch noch Statusabfragen getätigt und 16-Bit-Daten übernommen werden müssen, gerät der PPI hoffnungslos ins Hintertreffen.
Als einfachster und wirkungsvollster Eingriff ergab sich, den Reset-Eingang des READY-Flipflops 74LS76 (Pin 3, /R) von Pin 6 des 74LS00 zu trennen und auf eine der beiden noch freien Port-B-Leitungen des PPI zu legen (Tabelle, /RDYR). Da sich der Digitalisierer während des Einlesens eines Bildstreifens das letzte Kommando merken kann, braucht dieses nun nicht mehr nach jedem Datentransfer gesendet werden.

Tabelle: So wird der 8255 mit dem Digitalisierer verbunden

PC/8255			Digitalisierer	
Bez.	Anschl.		Bez.	Anschl.
PA0	33	<--->	D0	20c
PA1	35	<--->	D1	20a
PA2	37	<--->	D2	21c
PA3	39	<--->	D3	21a
PA4	41	<--->	D4	22c
PA5	43	<--->	D5	22a
PA6	45	<--->	D6	23c
PA7	47	<--->	D7	23a
PB0	15	--->	A0	12c
PB1	13	--->	A1	12a
PB2	11	--->	A2	11c
PB3	9	--->	S0(RD)	1a
PB4	7	--->	S1(WR)	2a
PB5	5	--->	S2(/IORQ)	1c
PB6	3	--->	/RDYR	LS76/Pin3
GND	2...50 (geradzahlig)	<--->	GND	32a, c

Statt dessen löscht man das READY-Signal direkt über die neu gelegte Verbindung /RDYR. Die Praxis hat gezeigt, daß so das Einlesen der Daten ohne Verluste erfolgen kann.

Parallel-Interface als Zugabe

Die kleine Interface-Platine ist schnell beschrieben. *Bild 2* zeigt ihre Schaltung. Sie ist reichlich überdimensioniert, um, nach der Devise: „Wenn schon, denn schon", auch andere Verwendungen zu erlauben. Für den Betrieb mit dem Digitalisierer braucht die Karte nicht unbedingt voll bestückt zu werden. Insbesondere kann dann der nicht benutzte PPI samt Treiber entfallen.
Der Vergleicher 74LS682 erkennt mit Unterstützung des Treibers 74LS244 und mehrerer NANDs I/O-Zugriffe auf einen bestimmten Speicherbereich. Sobald sich die Karte durch eine derartige Operation angesprochen fühlt, tritt der Decoder 74LS138 in Aktion und untersucht, wohin der Zugriff weitergeleitet werden soll. Dazu verwendet er die Adreßleitungen A2, A3 und A4, so daß acht aufeinanderfolgende Blöcke zu je vier Adressen zur Verfügung stehen. Die Lage dieser Blöcke im gesamten Adreßraum und ihre Zuordnung zu den beiden PPIs sind per Jumper einstellbar.
Da die Kanäle eines 8255 je nach Programmierung Ein- oder Ausgänge sein können, muß die Richtung der Treiber ebenfalls einstellbar sein. Hierfür sind wiederum einige Jumper vorgesehen. Beim Anschluß des Digitalisierers muß berücksichtigt werden, daß einer der benötigten 8-Bit-Ports, nämlich der, welcher den Datenbus bedient, abwechselnd als Ein- oder Ausgang fungiert. Der zugehörige Treiber sollte daher entfernt und überbrückt werden. Da im zweiten Port, der ständig auf Ausgabe geschaltet ist, noch ein Bit frei ist, wäre auch an eine softwaremäßige Richtungsumschaltung des Treibers zu denken. Aber Vorsicht! Es dürfen sich zu keiner Zeit 8255- und 74LS245-Ausgänge „gegenüberstehen".

Programm mit Komfort

Bild 3 zeigt das Digitalisierprogramm. Es wurde auf einen M24 von Olivetti mit CGA-Karte entwickelt. Ziel der Programmierung war es, ein schnelles Datenübernahmewerkzeug zu schaffen, das außerdem sofort die ankommenden Daten so gut wie möglich auf Bildschirm und Drucker sichtbar macht. Zusätzlich sollten die Bilder auch noch auf Diskette oder Festplatte geschrieben und von dort auch wieder gelesen werden können. Das ist vor allem dann wichtig, wenn Sie die Bilddaten per Programm weiterverarbeiten wollen und das Ergebnis auch wieder vor Augen haben möchten.

Die Empfangsroutine ist im Programm unter dem Label VIDI zu finden. Sie besteht hauptsächlich aus IN- und OUT-Befehlen. VIDI holt die drei den Grauebenen entsprechenden Teilbilder nacheinander streifenweise beim PPI ab und legt sie einfach in den Speicher, begin-

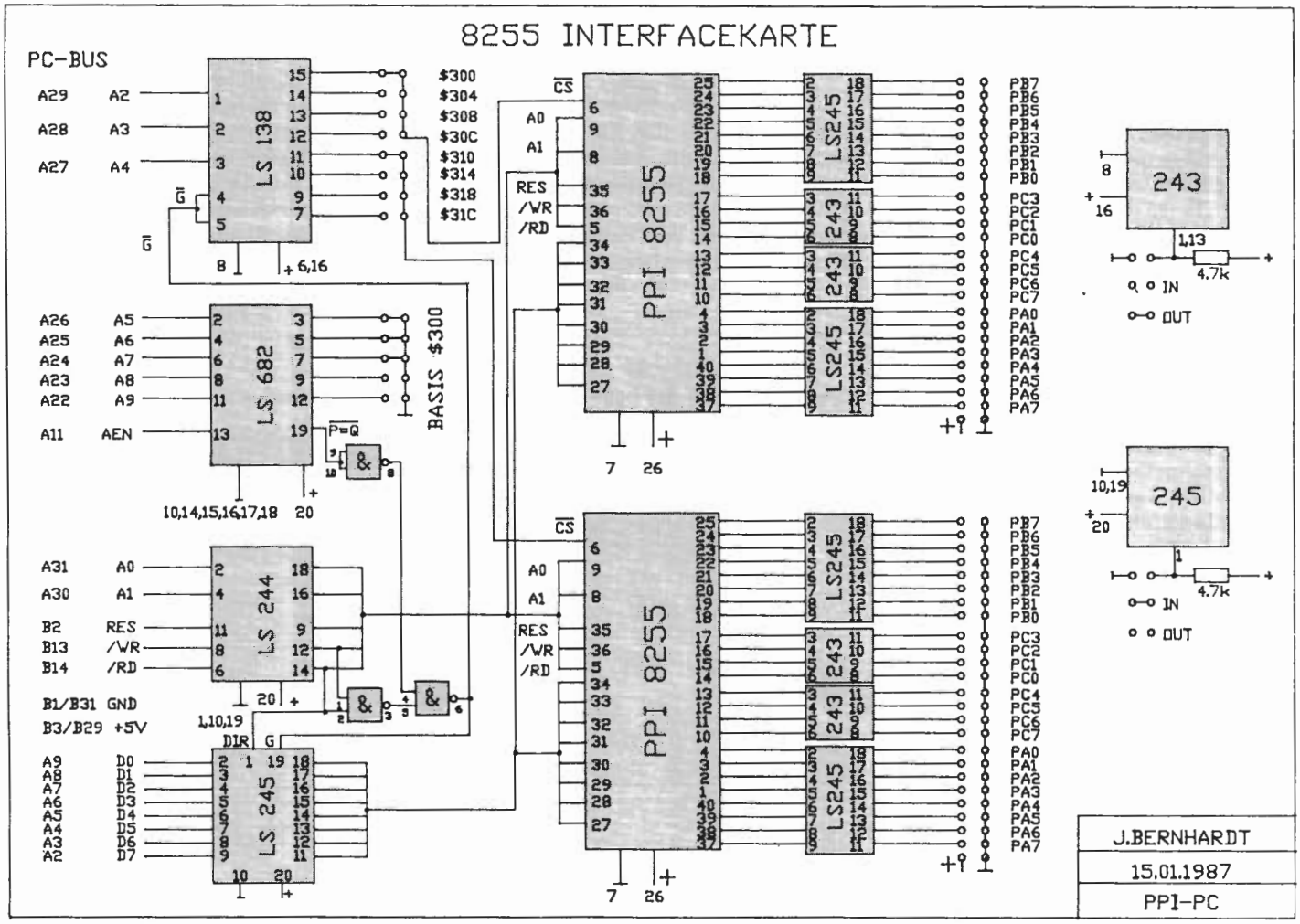


Bild 2. Mit dieser Schaltung läßt sich nicht nur der Digitalisierer ansteuern, sie eignet sich auch für zahlreiche andere Aufgaben

nend mit der am wenigsten signifikanten Grauebene.
Beim Bildaufbau auf dem Schirm waren leider Kompromisse notwendig. Denn das Bildformat 256 × 256 bei acht Graustufen schafft keiner der Grafikmodi des Farbgrafikadapters (CGA). Schon die acht Graustufen bereiten Schwierigkeiten: Der CGA gibt maximal vier her, die Auflösung sinkt dabei auf 320 × 200 Punkte.
Um nicht nur langweilige Schwarzweiß-Bilder zu produzieren, wählten wir diesen Grafikmodus. Da nun aber 200 Zeilen zu wenig sind und bei diesem Format, bedingt durch die Pixelverteilung des Digitalisierers, die schönen Bilder fürchterlich in die Länge gezogen würden, empfahl es sich, nur jede zweite Zeile darzustellen. Zu unserem eigenen Erstaunen erschienen dadurch auf Anhieb Kreise tatsächlich als Kreise, ohne weitere Tricks.
Die Routine DISPLAY besorgt die CGA-gerechte Überarbeitung der von VIDI abgespeicherten Bilddaten. Dazu muß man wissen, daß der Grafikspeicher des CGA ab Adresse B800:0000 beginnt und daß er recht merkwürdig aufgeteilt ist. Die einzelnen Bildzeilen liegen nämlich nicht hintereinander im Speicher: Ab B800:0000 aufwärts findet man nur die Zeilen mit gerader Nummer, die ungeradzahligen hingegen werden an der Adresse B800:2000 erwartet.
Aber ansonsten gibt es keine Besonderheiten. Die Bildpunkte einer Zeile liegen in jedem Byte hintereinander, die beiden Bits je Pixel Seite an Seite. DISPLAY bereitet stets acht aufeinanderfolgende Punkte vor (16 Bit) und schreibt sie auf einen Schlag in den Bildspeicher.
Die Umschaltung des CGA zwischen Text- und Grafikmodus erfolgt durch ANSI-Steuersequenzen. Das bedeutet, daß der Treiber ANSI.SYS in der Datei CONFIG.SYS eingetragen sein muß. Natürlich geht es auch anders: Wer will, kann auch einen Aufruf des Video-Interrupts INT 10H verwenden. In diesem Fall sollte die Routine SETMODE abgeändert werden.

Ausgeben mit 24-Nadel-Drucker

Selbstverständlich haben PCs heutzutage eine Bildschirm-Hardcopy-Funktion. Probieren Sie sie ruhig einmal aus! Wenn Sie alles richtig gemacht haben, werden sie, vorausgesetzt, ein IBM-kompatibler Drucker ist angeschlossen, auf dem Papier zwar schon etwas erhalten, was dem Bild auf Ihrem Schirm zumindest ähnlich sieht. Aber der wahre Genuß ist es nicht, insbesondere, weil es dummerweise „negativ" aus dem Drukker kommt. Bei Schriftzeichen und Grafiken ist das durchaus in Ordnung, aber Bilder sehen so recht merkwürdig aus. Außerdem fehlen auf diesen Hardcopies 75 % der ursprünglich zur Verfügung stehenden Datenmenge. Denn erstens wurde ja zur Darstellung eines verzerrungsfreien Bildes mittels CGA jede zweite Zeile weggelassen, zweitens kann der CGA nur vier von einst acht Graustufen reproduzieren. Warum also auf wertvolle Bildinformation verzichten?
Der Wunsch, auch dieses Problem noch zu lösen, führte zu der Routine namens PLOT. Da die Bilddaten in Streifen zu je 16 Pixels im Speicher liegen, ist die Aufbereitung für die bei Matrixdruckern übliche zeilenweise Verarbeitung ein wenig trickreich. Sie erfolgt über die Routine PLOTPREP, die im Zeilendatenpuffer PLOBUF stets 16 aufeinanderfolgende Bildspalten vorbereitet.
Um den Drucker nicht allzu lange mit einem einzigen Bild zu beschäftigen, haben wir großen Wert darauf gelegt, daß alle seine Drucknadeln im Einsatz sind. Der Pinwriter P6, für den die hier abgedruckte Programmversion eingerichtet ist, hat gleich 24 Stück davon zu bieten. Trotz allem erfolgt die Druckausgabe immer noch recht gemächlich, weil Plotdaten eben auch in gepackter Form noch sehr umfangreich sind.
Das Hauptproblem ist jedoch die saubere Darstellung der acht verschiedenen Graustufen. Um auf dem Papier ein verzerrungsfreies Bild zu bekommen, muß man zunächst ein wenig rechnen und probieren. Wir fanden heraus, daß beim P6, der im hier verwendeten Grafikmodus horizontal 360 und vertikal 180 Punkte pro Zoll schafft, ein Pixel als Punktmatrix zu je drei Zeilen und zehn Spalten gedruckt werden sollte. Man erhält dann einen ordentlich proportionierten Ausdruck (*Bild 4*).
Diese dreißig Punkte je Pixel können nun beliebige Muster enthalten, die den einzelnen Graustufen zugeordnet werden. Einige Tests ergaben, daß senkrechte Streifen, die um so schmaler werden, je heller der Bildpunkt ist, den besten Kontrast liefern (siehe Label PLOPAT am Ende des Programms). Andere Muster, wie z. B. Pünktchen in verschiedenen Abständen, Schraffierungen oder Karos taugen nichts, da die meisten Nadeldrucker die schwarzen Punkte viel zu dick hinklecksen, so daß dazwischenliegende weiße Punkte kaum noch sichtbar sind. Derart produzierte Bilder sehen dann ziemlich düster aus.
Die Drucker-Steuersequenzen befinden sich ganz am Ende des Programms. Dort sind vier Strings zu finden: PLOINIT und PLOEXIT werden je Bild genau einmal zu Anfang und am Schluß gesendet, PLOSTART und PLOSTOP erhält der Drucker vor bzw. nach jeder Druckzeile.

PLOSTOP hat die Aufgabe, das Papier in die richtige Stellung für die nächste Zeile zu schieben. PLOEXIT bringt den Drucker wieder in den Originalzustand und sorgt außerdem dafür, daß das Pa-

```
;-------------------------------------------------------------------------
;
; SBS DIGITAL PICTURE RECEIVER - VIDI Version 1.0
;
; Changes since version 0.0: Hardcopy feature added (NEC Pinwriter P6)
;
; (c) 1987 MicroSBS
; March 9th 1987
; Sven B. Schreiber
;
;-------------------------------------------------------------------------
;
CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
;
;-------------------------------------------------------------------------
;
DFCB    EQU     005CH           ;DEFAULT FILE CONTROL BLOCK
DBUF    EQU     0080H           ;DEFAULT RECORD BUFFER
TPA     EQU     0100H           ;TRANSIENT PROGRAM AREA
;
VIDBASE EQU     0B800H          ;VIDEO RAM PARAGRAPH
EVENOFF EQU     0000H           ;OFFSET OF EVEN VIDEO ROWS
ODDOFF  EQU     2000H           ;OFFSET OF ODD VIDEO ROWS
;
PPI1    EQU     0100H           ;8255 BASE ADDR (A-B-C-X)
;
NROWS   EQU     0100H           ;NUMBER OF PICTURE ROWS (MAX 256)
NCOLS   EQU     0100H           ;NUMBER OF PICTURE COLUMNS
NBITS   EQU     3               ;NUMBER OF BITS PER PICTURE PIXEL
VROWS   EQU     200             ;NUMBER OF VIDEO ROWS
```

Bild 3. Das Digitalisierprogramm: Datenübernahme und Bildaufbau dauern weniger als eine Sekunde

```
VCOLS   EQU     320             ;NUMBER OF VIDEO COLUMNS
DROWS   EQU     3               ;NUMBER OF DOT ROWS
DCOLS   EQU     10              ;NUMBER OF DOT COLUMNS
;
PICBUFS EQU     NBITS*NROWS*(NCOLS/8)   ;PICTURE BUFFER SIZE
PLOBUFS EQU     3*DCOLS*16              ;PLOT BUFFER SIZE
RECSIZE EQU     80H                     ;DOS RECORD SIZE
;
DOS     EQU     21H             ;DOS LIBRARY INTERRUPT #
;
EXITF   EQU     00H             ;EXIT PROGRAM
CONOUTF EQU     02H             ;CONSOLE OUTPUT
LISTF   EQU     05H             ;LIST OUTPUT
PRSTRF  EQU     09H             ;PRINT STRING
OPENF   EQU     0FH             ;OPEN FILE
CLOSEF  EQU     10H             ;CLOSE FILE
RDSEQF  EQU     14H             ;READ SEQUENTIAL
WRSEQF  EQU     15H             ;WRITE SEQUENTIAL
MAKEF   EQU     16H             ;MAKE FILE
SETDMAF EQU     1AH             ;SET DMA ADDRESS
;
EOT     EQU     04H             ;END OF TRANSMISSION
BEL     EQU     07H             ;BELL
LF      EQU     0AH             ;LINE FEED
FF      EQU     0CH             ;FORM FEED
CR      EQU     0DH             ;CARRIAGE RETURN
SUBST   EQU     1AH             ;SUBSTITUTE
ESC     EQU     1BH             ;ESCAPE
FS      EQU     1CH             ;FILE SEPARATOR
;
;-------------------------------------------------------------------------
;
        ORG     TPA
START:
        JMP     MAIN
        DW      FREEMEM
COPYRIGHT:
        DB      '                '
        DB      '                '
        DB      '----------------'
        DB      'SBS VIDI -- V1.0'
        DB      '(c)1987 MicroSBS'
        DB      '----------------'
        DB      '                '
        DB      'S. B. Schreiber '
STACK:
;
;-------------------------------------------------------------------------
;
MAIN:
        MOV     WORD PTR TOS,SP         ;SAVE STACK POINTER
        MOV     SP,OFFSET STACK         ;USE LOCAL STACK
        MOV     DX,OFFSET SBSMSG        ;PRINT MESSAGE
        CALL    PRSTR
        CALL    FILE
        MOV     DX,OFFSET COMMSG        ;PRINT COMMAND MESSAGE
        CALL    PRSTR
        TEST    BYTE PTR RDFLG,0FFH     ;CHECK IF READ MODE
        MOV     DX,OFFSET RCOMMSG
        JNZ     MAIN1
        TEST    BYTE PTR WRFLG,0FFH     ;CHECK IF WRITE MODE
        MOV     DX,OFFSET WCOMMSG
        JZ      MAIN2
MAIN1:
        CALL    PRSTR                   ;PRINT R/W MESSAGE
MAIN2:
        MOV     BYTE PTR PICFLG,0       ;RESET PICTURE FLAG
        MOV     BYTE PTR HALFFLG,0FFH   ;SET HALF HEIGHT FLAG
```

```
GETCOM:
        MOV     DX,OFFSET BELLSTR       ;PROMPT SIGNAL
        CALL    PRSTR
        CALL    CONIN                   ;GET COMMAND
        CMP     AL,ESC
        JZ      EXIT                    ;EXIT PROGRAM
        CMP     AL,' '
        JZ      RCVPIC                  ;RECEIVE PICTURE
        CMP     AL,'D'
        JZ      PLPIC                   ;PLOT PICTURE
        CMP     AL,'L'
        JZ      RDPIC                   ;READ PICTURE FROM FILE
        CMP     AL,'S'
        JZ      WRPIC                   ;WRITE PICTURE TO FILE
        CMP     AL,CR
        JZ      HEIGHT                  ;CHANGE HEIGHT
        JMP     GETCOM                  ;INVALID COMMAND
RCVPIC:
        MOV     BX,OFFSET GRASEQ        ;SET GRAPHICS MODE / CLEAR PAGE
        MOV     AH,CONOUTF
        CALL    SETMODE
        CALL    VIDI                    ;RECEIVE PICTURE
SHOWPIC:
        CALL    DISPLAY                 ;SHOW PICTURE
        MOV     BYTE PTR PICFLG,0FFH    ;SET PICTURE FLAG
        JMP     GETCOM
PLPIC:
        TEST    BYTE PTR PICFLG,0FFH    ;IGNORE COMMAND IF NO PICTURE LOADED
        JZ      GETCOM
        CALL    PLOT                    ;PLOT PICTURE
        JMP     GETCOM                  ;GET NEXT COMMAND
RDPIC:
        TEST    BYTE PTR RDFLG,0FFH     ;IGNORE IF NOT READ MODE
        JZ      GETCOM
        MOV     BX,OFFSET GRASEQ        ;SET GRAPHICS MODE / CLEAR PAGE
        MOV     AH,CONOUTF
        CALL    SETMODE
        CALL    READ                    ;READ PICTURE
        JMP     SHOWPIC                 ;SHOW PICTURE
WRPIC:
        TEST    BYTE PTR WRFLG,0FFH     ;IGNORE IF NOT WRITE MODE
        JZ      GETCOM
        TEST    BYTE PTR PICFLG,0FFH    ;IGNORE COMMAND IF NO PICTURE LOADED
        JZ      GETCOM
        CALL    WRITE                   ;WRITE PICTURE
        JMP     GETCOM                  ;GET NEXT COMMAND
HEIGHT:
        TEST    BYTE PTR PICFLG,0FFH    ;IGNORE COMMAND IF NO PICTURE LOADED
        JZ      GETCOM
        MOV     BX,OFFSET GRASEQ        ;SET GRAPHICS MODE / CLEAR PAGE
        MOV     AH,CONOUTF
        CALL    SETMODE
        NOT     BYTE PTR HALFFLG        ;INVERT HALF HEIGHT FLAG
        JMP     SHOWPIC                 ;DISPLAY SAME PICTURE
EXIT:
        MOV     DX,OFFSET ENDMSG        ;FINAL MESSAGE
XERROR:
        PUSH    DX
        MOV     BX,OFFSET TXTSEQ        ;SET TEXT MODE
        MOV     AH,CONOUTF
        CALL    SETMODE
        POP     DX                      ;PRINT ERROR MESSAGE
        CALL    PRSTR
        MOV     DX,OFFSET SBSMSG        ;PRINT PROGRAM MESSAGE
ERROR:
        CALL    PRSTR
        TEST    BYTE PTR WRFLG,0FFH     ;CLOSE TARGET FILE IF WRITE MODE
        JZ      ERROR1
        CALL    CLOSE
```

```
ERROR1:
        MOV     SP,WORD PTR TOS         ;RESTORE STACK POINTER
        MOV     AH,EXITF
        INT     DOS
;
;-------------------------------------------------------------------------
;
VIDI:
        MOV     DX,PPI1                 ;BUS INTERFACE PPI ADDR
        MOV     BX,OFFSET PICBUF        ;PICTURE DATA BUFFER
        MOV     CL,00000000B            ;START AT COLUMN 0, BIT 0
VIDI1:
        ADD     DX,3                    ;SET A & B TO OUTPUT, C TO INPUT
        MOV     AL,89H
        OUT     DX,AL
        SUB     DX,2                    ;SET -IORQ & WR, ADDR 3, RESET RDY
        MOV     AL,00010011B
        OUT     DX,AL
        DEC     DX                      ;SELECT COLUMN/BIT
        MOV     AL,CL
        OUT     DX,AL
        ADD     DX,3                    ;SET A & C TO INPUT, B TO OUTPUT
        MOV     AL,99H
        OUT     DX,AL
        SUB     DX,2                    ;SET -IORQ & RD, ADDR 2, ENABLE RDY
        MOV     AL,01001010B
        OUT     DX,AL
        DEC     DX
VIDI2:
        IN      AL,DX                   ;WAIT FOR VSYNC ACTIVE
        TEST    AL,00000010B
        JZ      VIDI2
        MOV     BP,200H                 ;LOAD TIME CONSTANT
VIDI3:
        SUB     BP,1                    ;IGNORE VERTICAL FRONT PORCH
        JNZ     VIDI3
        INC     DX
        MOV     AL,00001010B            ;RESET RDY
        OUT     DX,AL
        MOV     AL,01001010B            ;ENABLE RDY
        OUT     DX,AL
        DEC     DX
VIDI4:
        IN      AL,DX                   ;WAIT FOR RDY
        TEST    AL,00000001B
        JZ      VIDI4
        INC     DX                      ;SET -IORQ & RD, ADDR 0
        MOV     AL,01001000B
        OUT     DX,AL
        DEC     DX                      ;READ DATA BYTE (PIXELS 0..7)
        IN      AL,DX
        MOV     AH,AL
        INC     DX                      ;SET -IORQ & RD, ADDR 1, RESET RDY
        MOV     AL,00001001B
        OUT     DX,AL
        DEC     DX                      ;READ DATA BYTE (PIXELS 8..F)
        IN      AL,DX
        NOT     AX
        MOV     WORD PTR [BX],AX        ;SAVE DATA WORD
        ADD     BX,2
        INC     DX                      ;SET -IORQ & RD, ADDR 2, ENABLE RDY
        MOV     AL,01001010B
        OUT     DX,AL
        DEC     DX
        DEC     CH                      ;COUNT DOWN ROWS
        JNZ     VIDI4                   ;NEXT ROW
        INC     CL                      ;NEXT COLUMN OR NEXT BIT
        CMP     CL,NBITS*10H
        JB      VIDI1
        RET
;
;-------------------------------------------------------------------------
;
DISPLAY:
        PUSH    ES
        MOV     BX,VIDBASE              ;SHIFT EXTRA SEGMENT TO VIDEO RAM
        MOV     ES,BX
        MOV     BX,OFFSET PICBUF+(NROWS*(NCOLS/8))      ;SOURCE POINTER
        MOV     BP,0AAAAH               ;PREPARE COLOR MASK
DISPLAY1:
        MOV     SI,EVENOFF              ;TARGET POINTER, EVEN ROWS
        MOV     DI,ODDOFF               ;TARGET POINTER, ODD ROWS
        MOV     CX,NCOLS/16             ;PREPARE COLUMN COUNT
DISPLAY2:
        PUSH    CX                      ;SAVE COLUMN COUNT
        PUSH    SI                      ;SAVE TARGET POINTERS
        PUSH    DI
        PUSH    BX
        XOR     CX,CX                   ;RESET ROW COUNT
DISPLAY3:
        PUSH    CX
        MOV     AX,WORD PTR [BX]        ;GET PIXEL ARRAY
        ADD     BX,2
        MOV     CL,2*8                  ;PREPARE SHIFT COUNT
        CMP     BP,0AAAAH               ;CHECK IF INITIAL COLOR MASK
        JNZ     DISPLAY4                ;DON'T CLEAR GRAPHICS WORD IF NOT
        MOV     WORD PTR ES:[SI],0      ;CLEAR GRAPHICS WORD #1
        MOV     WORD PTR ES:[SI+2],0    ;CLEAR GRAPHICS WORD #2
DISPLAY4:
        ADD     AX,AX                   ;SHIFT NEXT PIXEL TO CARRY
        PUSHF
        RCL     DX,1                    ;SHIFT PIXEL TO DX (TWICE)
        POPF
        RCL     DX,1
        DEC     CL                      ;COUNT DOWN SHIFTS
        TEST    CL,07H
        JNZ     DISPLAY4
        XCHG    DH,DL                   ;SWAP LO & HI BYTES
        AND     DX,BP                   ;APPLY COLOR MASK
        OR      WORD PTR ES:[SI],DX     ;WRITE GRAPHICS WORD
        ADD     SI,2
        OR      CL,CL                   ;CHECK IF PIXEL ARRAY WRITTEN
        JNZ     DISPLAY4
        ADD     SI,(VCOLS/4)-4          ;SET POINTER TO NEXT ROW
        XCHG    SI,DI                   ;CHANGE TARGET POINTER (EVEN <--> ODD)
        POP     CX
        MOV     AX,VROWS
        TEST    BYTE PTR HALFFLG,0FFH   ;CHECK IF HALF HEIGHT MODE
        JZ      DISPLAY5                ;FULL HEIGHT
        INC     CX                      ;SKIP ROW NUMBER
        CMP     CX,NROWS
        JNB     DISPLAY6                ;COLUMN WRITTEN
        ADD     BX,2                    ;SKIP DATA OF NEXT ROW
        ADD     AX,AX
DISPLAY5:
        INC     CX                      ;COUNT ROWS
        CMP     CX,AX
        JNB     DISPLAY6                ;OUTSIDE DISPLAY AREA
        CMP     CX,NROWS
        JB      DISPLAY3                ;NEXT ROW
DISPLAY6:
        POP     BX                      ;RESTORE SOURCE POINTER
        POP     DI                      ;RESTORE TARGET POINTERS
        POP     SI
        POP     CX                      ;RESTORE COLUMN COUNT
        ADD     BX,NROWS*2              ;SET SOURCE POINTER TO NEXT COLUMN
        ADD     SI,4                    ;SET TARGET POINTERS TO NEXT COLUMN
        ADD     DI,4
```

```
        LOOP    DISPLAY2                ;NEXT COLUMN
        NOT     BP                      ;CHANGE COLOR MASK
        CMP     BP,0AAAAH               ;CHECK IF BOTH COLOR MASKS APPLIED
        JZ      DISPLAY7                ;READY
        JMP     DISPLAY1                ;NEXT COLOR BIT
DISPLAY7:
        POP     ES                      ;SHIFT BACK EXTRA SEGMENT
        RET
;
;-------------------------------------------------------------------------
;
PLOT:
        MOV     WORD PTR PLOCNT,0       ;RESET PLOT COUNT
        MOV     BX,OFFSET PLOINIT       ;ENTER PLOT MODE
        MOV     AH,LISTF
        CALL    SETMODE
        MOV     BX,OFFSET PICBUF        ;SOURCE ADDRESS
PLOT1:
        PUSH    BX
        MOV     BX,OFFSET PLOSTART      ;START PLOTTING
        MOV     AH,LISTF
        CALL    SETMODE
        POP     BX
        PUSH    BX
        MOV     CX,NCOLS/16             ;COLUMN WORD COUNT
PLOT2:
        PUSH    CX
        PUSH    BX
        MOV     AX,WORD PTR PLOCNT      ;SET AUXILIARY PLOT COUNT
        MOV     WORD PTR PLOCNT1,AX
        CALL    PLOTPREP                ;PREPARE NEXT BLOCK
        MOV     DI,OFFSET PLOBUF
        MOV     CX,PLOBUFS              ;PLOT DATA COUNT
PLOT3:
        PUSH    DI
        PUSH    CX
        MOV     DL,BYTE PTR [DI]        ;SEND PLOT DATA BYTE
        MOV     AH,LISTF
        INT     DOS
        POP     CX
        POP     DI
        INC     DI                      ;POINTER TO NEXT PLOT DATA BYTE
        LOOP    PLOT3
        POP     BX
        ADD     BX,NROWS*2              ;POINTER TO NEXT COLUMN WORD
        POP     CX
        LOOP    PLOT2
        MOV     BX,OFFSET PLOSTOP       ;STOP PLOTTING
        MOV     AH,LISTF
        CALL    SETMODE
        POP     BX
        ADD     BX,(24/DROWS)*2         ;POINTER TO NEXT ROW GROUP
        ADD     WORD PTR PLOCNT,24/DROWS
        CMP     WORD PTR PLOCNT,NROWS
        JB      PLOT1
        MOV     BX,OFFSET PLOEXIT       ;EXIT PLOT MODE
        MOV     AH,LISTF
        JMP     SETMODE
;
;-------------------------------------------------------------------------
;
PLOTPREP:
        MOV     DI,OFFSET PLOBUF        ;TARGET: PLOT BUFFER
        MOV     CX,24/DROWS             ;ROW COUNT
PLOTPREP1:
        PUSH    CX
        MOV     BP,PICBUFS/NBITS        ;SIZE OF PARTIAL PICTURE
        MOV     SI,BP                   ;POINTER TO PARTIAL PIC, GRAY LEVEL #1
        ADD     SI,BX
        ADD     BP,BP                   ;POINTER TO PARTIAL PIC, GRAY LEVEL #2
        ADD     BP,BX
        PUSH    BX
        PUSH    SI
        PUSH    BP
        PUSH    DI
        MOV     BX,WORD PTR [BX]        ;GET COLUMN WORD, GRAY LEVEL #0
        MOV     DX,WORD PTR [SI]        ;GET COLUMN WORD, GRAY LEVEL #1
        MOV     BP,WORD PTR [BP]        ;GET COLUMN WORD, GRAY LEVEL #2
        MOV     CX,16                   ;BIT COUNT
PLOTPREP2:
        PUSH    CX
        XOR     AL,AL                   ;CLEAR TARGET REGISTER
        ADD     BP,BP                   ;SHIFT LEFT GRAY LEVEL WORD #2
        ADC     AL,AL                   ;SHIFT CARRY LEFT INTO TARGET REGISTER
        ADD     DX,DX                   ;SHIFT LEFT GRAY LEVEL WORD #1
        ADC     AL,AL                   ;SHIFT CARRY LEFT INTO TARGET REGISTER
        ADD     BX,BX                   ;SHIFT LEFT GRAY LEVEL WORD #0
        ADC     AL,AL                   ;SHIFT CARRY LEFT INTO TARGET REGISTER
        MOV     CX,DCOLS                ;COMPUTE ADDRESS OF GRAY PATTERN
        MUL     CL
        MOV     SI,OFFSET PLOPAT
        ADD     SI,AX
PLOTPREP3:
        MOV     AL,BYTE PTR [SI]        ;GET PATTERN BYTE
        MOV     AH,DROWS                ;DOT ROW COUNT
        CMP     WORD PTR PLOCNT1,NROWS
        JB      PLOTPREP4
        XOR     AL,AL                   ;REMAINING BOTTOM ROWS ARE BLANK
PLOTPREP4:
        RCR     AL,1                    ;SHIFT PATTERN TO TARGET WORD
        RCL     BYTE PTR [DI+2],1
        RCL     BYTE PTR [DI+1],1
        RCL     BYTE PTR [DI],1
        DEC     AH
        JNZ     PLOTPREP4
        INC     SI                      ;POINTER TO NEXT PATTERN BYTE
        ADD     DI,3                    ;POINTER TO NEXT TARGET WORD
        LOOP    PLOTPREP3
        POP     CX                      ;COUNT DOWN BITS
        LOOP    PLOTPREP2
        POP     DI
        POP     BP
        POP     SI
        POP     BX
        ADD     BX,2                    ;POINTER TO NEXT ROW
        INC     WORD PTR PLOCNT1
        POP     CX                      ;COUNT DOWN ROWS
        LOOP    PLOTPREP1
        MOV     CX,24 MOD DROWS
        JCXZ    PLOTPREP7
        MOV     AH,PLOBUFS/3
PLOTPREP5:
        PUSH    CX
PLOTPREP6:
        OR      AL,AL                   ;RESET CARRY
        RCL     BYTE PTR [DI+2],1       ;ADJUST PLOT BUFFER
        RCL     BYTE PTR [DI+1],1
        RCL     BYTE PTR [DI],1
        LOOP    PLOTPREP6
        POP     CX
        ADD     DI,3
        DEC     AH
        JNZ     PLOTPREP5
PLOTPREP7:
        RET
;
;-------------------------------------------------------------------------
```

```
;
SETMODE:
        MOV     CL,BYTE PTR [BX]        ;GET LENGTH BYTE
        XOR     CH,CH
SETMODE1:
        INC     BX
        MOV     DL,BYTE PTR [BX]        ;GET CONTROL BYTE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        INT     DOS                     ;SEND CONTROL BYTE
        POP     CX
        POP     BX
        POP     AX
        LOOP    SETMODE1
        RET
;
;-------------------------------------------------------------------
;
FILE:
        MOV     BYTE PTR RDFLG,0        ;RESET READ FLAG
        MOV     BYTE PTR WRFLG,0        ;RESET WRITE FLAG
        MOV     BX,OFFSET DFCB          ;CHECK FILE NAME IN FCB
        MOV     AH,BYTE PTR [BX+1]
        MOV     AL,BYTE PTR [BX+1+8]
        CMP     AX,'  '
        JNZ     FILE1
        MOV     DX,OFFSET INFOMSG       ;PRINT INFO MESSAGE
        JMP     PRSTR
FILE1:
        MOV     AL,BYTE PTR [BX+10H+1]  ;CHECK MODE ID
        CMP     AL,'L'
        JZ      SETREAD                 ;READ MODE
        CMP     AL,'S'
        JZ      SETWRITE                ;WRITE MODE
        MOV     DX,OFFSET MODERM        ;INVALID MODE ID
        JMP     ERROR
SETREAD:
        MOV     BYTE PTR RDFLG,0FFH     ;SET READ FLAG
        JMP     OPEN                    ;OPEN SOURCE FILE
SETWRITE:
        MOV     BYTE PTR WRFLG,0FFH     ;SET WRITE FLAG
        JMP     MAKE                    ;MAKE TARGET FILE
;
;-------------------------------------------------------------------
;
OPENFILE:
        MOV     BX,DX
        MOV     BYTE PTR [BX+0CH],0     ;SELECT EXTENT #0
        MOV     BYTE PTR [BX+20H],0     ;SELECT RECORD #0
        MOV     AH,OPENF                ;OPEN FILE
        INT     DOS
        INC     AL
        RET
OPEN:
        MOV     BYTE PTR EOFFLG,0       ;RESET END-OF-FILE FLAG
        MOV     DX,OFFSET DFCB          ;OPEN FILE
        CALL    OPENFILE
        JNZ     OPEN1
        MOV     DX,OFFSET OPENERM       ;SOURCE FILE NOT FOUND
        JMP     ERROR
OPEN1:
        RET
MAKE:
        MOV     DX,OFFSET DFCB          ;CHECK IF TARGET FILE EXISTS
        CALL    OPENFILE
        JZ      MAKE1
        MOV     DX,OFFSET DUPERM        ;DUPLICATE FILE NAME
        JMP     ERROR
MAKE1:
        MOV     DX,OFFSET DFCB          ;MAKE FILE
        MOV     AH,MAKEF
        INT     DOS
        INC     AL
        JNZ     MAKE2
        MOV     DX,OFFSET MAKERM        ;DIRECTORY FULL
        JMP     ERROR
MAKE2:
        RET
SETDBUF:
        MOV     DX,DBUF                 ;SELECT DEFAULT RECORD BUFFER
        MOV     AH,SETDMAF
        INT     DOS
        RET
CLOSE:
        MOV     DX,OFFSET DFCB          ;CLOSE TARGET FILE
        MOV     AH,CLOSEF
        INT     DOS
        RET
;
;-------------------------------------------------------------------
;
READ:
        MOV     DX,OFFSET DBUF          ;SELECT BUDDER ADDR
        MOV     AH,SETDMAF
        INT     DOS
        MOV     DI,OFFSET PICBUF        ;LOAD ADDR
        MOV     CX,PICBUFS              ;BYTE COUNT
        PUSH    CX
        CLD                             ;SELECT LOAD DIRECTION
READ1:
        PUSH    DI
        MOV     SI,OFFSET DBUF          ;LOAD PATTERN TO BUFFER
        MOV     DI,OFFSET DBUF+1
        MOV     CX,RECSIZE-1
        MOV     BYTE PTR [SI],0FH
        REP     MOVSB
        TEST    BYTE PTR EOFFLG,0FFH    ;DON'T LOAD RECORD IF END OF FILE
        JNZ     READ2
        MOV     DX,OFFSET DFCB          ;LOAD RECORD
        MOV     AH,RDSEQF
        INT     DOS
        MOV     BYTE PTR EOFFLG,AL      ;SET FLAG IF END OF FILE
READ2:
        POP     DI
        POP     CX
        SUB     CX,RECSIZE              ;COUNT DOWN BYTES
        JZ      READ3                   ;LAST RECORD
        JB      READ3                   ;LAST RECORD
        PUSH    CX
        MOV     CX,RECSIZE              ;COPY LOADED RECORD
        MOV     SI,OFFSET DBUF
        REP     MOVSB
        JMP     READ1                   ;NEXT RECORD
READ3:
        ADD     CX,RECSIZE              ;PREPARE BYTE COUNT FOR LAST RECORD
        MOV     SI,OFFSET DBUF          ;COPY REMAINING BYTES
        REP     MOVSB
        RET                             ;READY
;
;-------------------------------------------------------------------
;
WRITE:
        MOV     DX,OFFSET PICBUF        ;PICTURE ADDR
WRITE1:
        PUSH    DX
        MOV     AH,SETDMAF              ;SELECT RECORD ADDR
        INT     DOS
```

```
        MOV     DX,OFFSET DFCB          ;SAVE RECORD
        MOV     AH,WRSEQF
        INT     DOS
        OR      AL,AL                   ;CHECK RETURN CODE
        JZ      WRITE2
        MOV     DX,OFFSET WRERM         ;DISK FULL
        JMP     XERROR
WRITE2:
        POP     DX
        ADD     DX,RECSIZE              ;COMPUTE NEXT RECORD ADDR
        CMP     DX,OFFSET PICBUF+PICBUFS
        JB      WRITE1                  ;STILL DATA
        RET
;
;-------------------------------------------------------------------------
;
PRSTR:
        MOV     AH,PRSTRF               ;PRINT STRING
        INT     DOS
        RET
CONIN:
        MOV     DL,0FFH                 ;CHECK CONSOLE STATUS / GET DATA
        MOV     AH,6
        INT     DOS
        OR      AL,AL
        JZ      CONIN                   ;WAIT FOR DATA
        CMP     AL,'a'                  ;CHECK IF LOWER CASE
        JB      CONIN1
        CMP     AL,'z'+1
        JNB     CONIN1
        SUB     AL,20H                  ;CONVERT TO UPPER CASE
CONIN1:
        RET
;
;-------------------------------------------------------------------------
;
SBSMSG:
        DB      CR,LF
        DB      'SBS DIGITALBILD-EMPFAENGER - VIDI Version 1.0',CR,LF
        DB      CR,LF
        DB      '(c) 1987 MicroSBS',CR,LF
        DB      '9. Maerz 1987',CR,LF
        DB      'Sven B. Schreiber',CR,LF,'$'
MODERM:
        DB      CR,LF
        DB      'Sie verwenden ein falsches Kommandoformat.'
        DB      CR,LF
INFOMSG:
        DB      CR,LF
        DB      'Falls Sie Bilddateien lesen bzw. schreiben wollen, sollten '
        DB      'Sie das Programm',CR,LF
        DB      'folgendermassen starten:',CR,LF
        DB      CR,LF
        DB      '     VIDI <Quelldatei> L     (zum Lesen)',CR,LF
        DB      '     VIDI <Zieldatei>  S     (zum Schreiben)',CR,LF,'$'
COMMSG:
        DB      CR,LF
        DB      'Folgende Kommandos stehen zur Verfuegung:',CR,LF
        DB      CR,LF
        DB      '     ESCAPE-TASTE  -  Programm verlassen',CR,LF
        DB      '     LEERTASTE     -  neues Bild digitalisieren',CR,LF
        DB      '     RETURN-TASTE  -  Bildhoehe umschalten',CR,LF
        DB      '     D             -  Bild drucken (NEC Pinwriter P6)'
        DB      CR,LF,'$'
RCOMMSG:
        DB      '     L             -  Bild aus Quelldatei lesen',CR,LF,'$'
WCOMMSG:
        DB      '     S             -  Bild in Zieldatei schreiben',CR,LF,'$'
ENDMSG:
        DB      CR,LF
        DB      'Ende des Programmlaufs.',CR,LF,'$'
OPENERM:
        DB      CR,LF
        DB      'Die angegebene Quelldatei existiert nicht.',CR,LF,'$'
DUPERM:
        DB      CR,LF
        DB      'Die angegebene Zieldatei existiert bereits.',CR,LF,'$'
MAKERM:
        DB      CR,LF
        DB      'Das Inhaltsverzeichnis der Zieldiskette ist voll.',CR,LF,'$'
WRERM:
        DB      CR,LF
        DB      'Die Zieldiskette ist voll.',CR,LF,'$'
BELLSTR:
        DB      BEL,'$'
;
;-------------------------------------------------------------------------
;
GRASEQ:
        DB      5                       ;SET GRAPHICS MODE
        DB      ESC,'[=4h'
TXTSEQ:
        DB      5                       ;SET TEXT MODE
        DB      ESC,'[=2h'
PLOINIT:
        DB      6                       ;ENTER PLOT MODE
        DB      FS,'@',CR,ESC,'3',24
PLOEXIT:
        DB      7                       ;EXIT PLOT MODE
        DB      ESC,'3',156,LF,LF,FS,'@'
PLOSTART:
        DB      4                       ;START PLOTTING
        DB      FS,'Z'
        DW      NCOLS*DCOLS
PLOSTOP:
        DB      2                       ;STOP PLOTTING
        DB      CR,LF
PLOPAT:
        DB      111B,111B,111B,111B,111B,111B,111B,111B,111B,111B       ;BLACK
        DB      000B,111B,111B,111B,111B,111B,111B,111B,111B,000B       ;GRAY 1
        DB      000B,111B,111B,111B,111B,111B,111B,111B,000B,000B       ;GRAY 2
        DB      000B,000B,111B,111B,111B,111B,111B,000B,000B,000B       ;GRAY 3
        DB      000B,000B,000B,111B,111B,111B,111B,000B,000B,000B       ;GRAY 4
        DB      000B,000B,000B,000B,111B,111B,000B,000B,000B,000B       ;GRAY 5
        DB      000B,000B,000B,000B 111B,000B,000B,000B,000B,000B       ;GRAY 6
        DB      000B,000B,000B,000B,000B,000B,000B,000B,000B,000B       ;WHITE
;
;-------------------------------------------------------------------------
;
TOS     EQU  $                          ;TOP OF STACK
HALFFLG EQU  TOS + 2                    ;HALF HEIGHT FLAG
RDFLG   EQU  HALFFLG + 1                ;READ FLAG
WRFLG   EQU  RDFLG + 1                  ;WRITE FLAG
EOFFLG  EQU  WRFLG + 1                  ;END-OF-FILE FLAG
PICFLG  EQU  EOFFLG + 1                 ;PICTURE FLAG
PLOCNT  EQU  PICFLG + 1                 ;PLOT COUNT
PLOCNT1 EQU  PLOCNT + 2                 ;PLOT COUNT (AUXILIARY)
PICBUF  EQU  PLOCNT1 + 2                ;PICTURE BUFFER
PLOBUF  EQU  PICBUF + PICBUFS           ;PLOT BUFFER
;
FREEMEM EQU  PLOBUF + PLOBUFS
;
;-------------------------------------------------------------------------
;
CODE    ENDS
        END  START
```

pier so eingestellt wird, daß exakt zwei Bilder auf eine Seite passen.

Bedienung sehr einfach

Das Programm läßt sich mit MASM 3.00 übersetzen und mit LINK 2.20 und EXE2BIN startfähig machen. Die bekannte Linker-Warnung „No Stack Segment" darf ignoriert werden. Die resultierende COM-Datei hat eine Größe von etwa 3 KByte und paßt somit wohl auf jede Diskette, die wenigstens ein bißchen Platz hat. Die Bilddateien, die das Programm produziert, benötigen 24 KByte je Bild.
Beim Aufruf des Programms sind drei verschiedene Kommandoformate möglich:

– VIDI
– VIDI ⟨Quelldatei⟩ L
– VIDI ⟨Zieldatei⟩ S

Der erste davon erlaubt lediglich Digitalisieren, Darstellen auf dem Schirm und Hardcopy von Bildern, ohne Dateiverwaltung. Aufrufnummer zwei ermöglicht das sequentielle Einlesen einer bereits bestehenden Bild-Quelldatei, Nummer drei das Ablegen mehrerer Bilder in die angegebene Zieldatei.

Abhängig vom verwendeten Kommandoformat stehen dem Benutzer vier oder fünf Kommandos zur Verfügung. Die Eingabebereitschaft signalisiert das Programm durch einen Piepser, da es den Bildschirm ja für andere Zwecke benötigt.

Die Escape-Taste ist besonders wichtig. Sie ermöglicht das Beenden des Programms. Drücken der Leertaste startet einen neuen Digitalisierungsprozeß mit anschließender visueller Darbietung des Ergebnisses. Die Eingabe des Buchstabens D bewirkt das Ausdrucken des gerade auf dem Schirm sichtbaren Bildes. Mit der RETURN-Taste kann das Bildformat eingestellt werden. Voreingestellt ist das vorher beschriebene, bei dem nur jede zweite Zeile sichtbar ist. Wer will, kann die fehlenden Zeilen einfügen lassen, das Bild wird dadurch allerdings verzerrt, außerdem fehlen am unteren Bildrand 56 Zeilen.

Ein Programmaufruf mit Dateiangabe spendiert dem Benutzer eine weitere Option. Je nachdem, ob hinter dem Dateinamen ein L oder ein S stand, kann er dann auch Dateien lesen (L) oder schreiben (S). Im ersten Fall erscheint nach dem Drücken von L jeweils das nächste in der Quelldatei vorgefundene Bild auf dem Schirm, im zweiten bewirkt die Taste S das Schreiben des gerade sichtbaren Bildes in die Zieldatei, so wie es vom Digitalisierer empfangen wurde.

Bild 4. Ein Beispiel dafür, was der Pinwriter P6 aus dem digitalisierten Bild macht (der Ausdruck füllt in der Breite etwa eine DIN-A4-Seite)

Jeder dieser beiden Modi hat einen Sonderfall: Ist die Quelldatei abgearbeitet, produziert jeder weitere Leseversuch ein Balkenmuster auf dem Bildschirm. Falls überhaupt kein Bild vorhanden ist, wie z. B. direkt nach dem Programmstart, werden eventuelle Schreibwünsche einfach ignoriert, ebenso wie auch Druckkommandos ohne Bild wirkungslos sind.

Gibt der Benutzer keinen Dateinamen an, informiert ihn das Programm über die möglichen Kommandoformate. Auch ein Neuling, der probeweise einfach VIDI tippt, weiß dann Bescheid. Ist ein Dateiname vorhanden, prüft das Programm, ob ein L oder S dahinter steht. Ist das nicht der Fall, steigt es aus, jedoch nicht ohne dem Benutzer mitzuteilen, was er falsch gemacht hat und wie er diesen Fehler beheben kann.

Literatur

[1] *Moraw, K.:* Digitalisieren mit der Kamera. mc 1985, Heft 10, Seite 46...54.

TERM1 mit 19200 Baud

Beim mc-CP/M-Computer läßt sich für die Datenübertragung von der OUT1-Karte zum mc-Grafikterminal standardmäßig nur eine maximale Baudrate von 9600 Bit/s einstellen. Wenn bei bewegter Grafik große Datenmengen schnell übertragen werden müssen, bildet dieser Baudratenwert den berüchtigten Flaschenhals, vor allem dann, wenn die CPU auf der SYS1-Karte bereits mit einer Taktfrequenz von 6 MHz arbeitet. Eine Verdoppelung der Baudrate auf 19 200 Bit/s kann die Situation deutlich entschärfen.

Dazu wird auf der TERM1-Platine der 7,3728-MHz-Quarz gegen einen anderen mit der Frequenz 4,9152 MHz ausgetauscht. Die Jumpereinstellung für 28 800 Baud mit der Standardversion liefert mit dem neuen Quarz die gewünschten 19 200 Baud.

OUT1 kann mit DIP-Schaltern auf die neue Baudrate (19 200 Bit/s extern) eingestellt werden. Die Anpassung einer Tastatur mit Parallelausgang, die über den in mc 8/1983 beschriebenen parallel-seriellen Wandler an TERM1 angeschlossen ist, erfolgt durch den Austausch des 5-kΩ-Trimmers gegen einen 10-kΩ-Trimmer. Damit kann die neue Baudrate problemlos eingestellt werden. Wem jetzt die Taktfrequenz für die TERM1-CPU zu niedrig geworden ist, der kann noch die Verbindung zum Pin 6 des Z80-Prozessors auftrennen und den Takt mit einer Brücke zum Pin c29 der VG-64-Steckerleiste verbinden.

B. Brüggemann

Ingrid Trommer

Einführung in Unix

Teil 1

Unix ist ein Betriebssystem, das zunehmend eingesetzt wird. mc gibt einen Einblick in Unix und erklärt Grundbegriffe dieses multiuser- und multitasking-fähigen Betriebssystems. Der erste Teil der Serie schildert die Geschichte von Unix und geht auf die verschiedenen Versionen ein. Der Kernel und das Dateisystem werden ausführlich beschrieben.

Zwei Aussagen über Unix kann man in der inzwischen vielfältigen Literatur immer wieder lesen.

„Heutzutage ist Unix eines der mächtigsten Betriebssysteme, das sowohl auf Großsystemen, Minicomputern und Mikros verfügbar ist" [1].

„Die Wahrheit über Unix: Die Benutzerschnittstelle ist fürchterlich" [2].

Welche der beiden Meinungen für den jeweils besonderen Fall zutrifft, wird in unserer Serie untersucht werden.

Ein Betriebssystem mit Geschichte

Die Unix-Entwicklung begann 1969 in den Bell-Laboratorien von AT&T. Sie ist eine der größten Forschungsorganisationen der Welt und hatte damals bereits über 23 000 Mitarbeiter. Die Gruppe um Ken Thompson war mit einem Projekt beschäftigt, das Space Travel genannt wurde. Mit diesem Programm sollten die Bewegungen der wichtigsten Himmelskörper des Sonnensystems simuliert werden. Dazu stand ein Rechner GE645 von General-Electric mit dem Betriebssystem Multics zur Verfügung. Multics, eine Entwicklung des MIT (Massachusets Institute of Technologie), war eines der ersten interaktiven Mehrbenutzer-Betriebssysteme. Während der Programmentwicklung stellte sich heraus, daß die Interaktion des einzelnen Benutzers mit der Anlage zu teuer wurde. Ken Thompson schrieb deshalb Space Travel neu für eine PDP7, die mehr oder weniger unbenutzt in der Ecke stand. Für die Systemprogrammierer war dies eine unbefriedigende Lösung, da es auf der PDP7 nicht möglich war, den Quellcode zu speichern. Änderungen mußten auf der GE645 editiert, compiliert und dann auf Lochstreifen gestanzt werden. Der Objektcode konnte dann in die PDP7 geladen und ausgeführt werden.
Um eine schnelleres und sichereres Verfahren zu finden, schrieb Ken Thompson ein Betriebssystem für die PDP7 in Assembler und nannte es Unix. Der Name Unix ist aus der Wortkombination „Multics" und „unified conception" entstanden. An mehreren Eigenschaften ist erkennbar, daß Unix von Programmierern für Programmierer geschrieben wurde:

- Vielfältige Software-Werkzeuge, die besonders die Programmentwicklung erleichtern und jede Art von Textmanipulation unterstützen.
- Die Offenheit des Systems: Grundsätzlich kann jeder mit den Dateien des anderen arbeiten.
- Die Syntax: Kennzeichen kurz, aber unverständlich. Da Programmierer lieber kreativ sind, als stur Zeichen über die Tastatur einzugeben, kommt ihnen die kurze Syntax sehr entgegen.

Unix erfreute sich in den Bell-Laboratorien bald großer Beliebtheit. Nachdem die Anerkennung so groß wurde, war der nächste logische Schritt, das System maschinenunabhängig – in einer höheren Programmiersprache – zu schreiben. Dazu entwickelte Ken Thompson unter Anlehnung an BCPL und in gewohnter Abkürzungsfreude die Programmiersprache B (BCPL wurde 1967 von Richards zum Compilerbau entwickelt). Da B kein Typenkonzept hatte, erwies sie sich als zu wenig mächtig. Dennis Ritchie modifizierte deshalb diese Programmiersprache und als Weiterentwicklung entstand C. Heute sind 90 % von Unix in C geschrieben. Damit konnte Unix verhältnismäßig einfach auf andere Rechner übertragen werden. Ab 1973 wurden von der AT&T-Tochter Western-Electric Unix-Lizenzen zu einem symbolischen Preis Verwaltungen und Universitäten zur Verfügung gestellt. Besonders die Universitäten entwickelten sich zu begeisterten Unix-Benutzern.

Die Unix-Welt

Für den Unix-Neuling ist zunächst einmal die Namensvielfalt verwirrend; denn aus lizenzrechtlichen Gründen darf nur das Unix, das von AT&T vertrieben wird, wirklich Unix heißen. In der Unix-Welt gibt es zwei Unterschiede:

- Betriebssysteme, die wie Unix aussehen (Unix look alikes)
- Unix-Portierungen

Bei einem „look alike" hat sich der Anbieter die Ausgabe für die Quell-Lizenz gespart und selbst ein Betriebssystem entwickelt, das entweder nur auf der Benutzeroberfläche oder auch auf einer tieferen Ebene wie Unix aussieht. Diese Systeme sind für den Benutzer oft billiger. Es werden oft auch im Funktionsumfang erweiterte Versionen angeboten (z. B. Echtzeitsysteme oder Mehrprozessorunterstützung). Wie gut es in solchen Fällen mit der vielgelobten Unix-Portabilität aussieht, ist oft recht schwierig festzustellen.
Bei einer Unix-Portierung wurde von dem jeweiligen Anbieter eine Unix-Lizenz von AT&T erworben und der Quellcode auf die jeweilige Hardware portiert. Auch der Benutzer muß hier eine Lizenzgebühr an AT&T entrichten. Bei Portierungen ist es wichtig zu wissen, auf welcher Version von AT&T diese beruht. Die wichtigsten Unix-Versionen:

Version 7 1979	Version V.2 1984
Version III 1981	Version V.3 1986

Jede dieser Versionen enthält gegenüber der vorhergehenden Durchsatzverbesserungen und/oder wesentliche Erweiterungen des Funktionsumfangs. Teilweise wurden auch Konzepte neu überdacht und völlig neu implementiert. Es ist also von der Version 7 zu dem System V keine Aufwärtskompatibilität gegeben. Weiter trägt zur Verwirrung bei, daß es in der Unix-Entwicklung neben dem AT&T-Zweig noch den Berkeley-Zweig

gibt. Die aktuellen Versionen dieser Hochschule heißen hier 4.X Bsd (Berkeley Software Distribution). Berkeley hat sehr viel in Richtung Benutzerfreundlichkeit getan. Es wurde z. B. ein bildschirmorientierter Editor entwickelt, der auf sehr unterschiedlichen Terminals lauffähig ist.
Die Versionen von Bell und Berkeley mit Versionsnummer und Entstehungsjahr sind aus *Bild 1* zu ersehen.
Die meisten Anbieter bekennen sich inzwischen zu dem AT&T-Standard.
AT&T System V ist wohl auch deswegen zu einem Defacto-Industrie-Standard geworden, weil das System von AT&T seit 1983 kommerziell betreut und gewartet wird. Oft werden zusätzlich sogenannte Berkeley-Erweiterungen mit angeboten.
Die Entwicklung von Bell und Berkeley streben nicht soweit auseinander, wie es hier vielleicht den Anschein hat. Beide haben Verbesserungen und Erweiterungen des jeweils anderen in der nächsten oder übernächsten Version mit aufgenommen.
Bild 2 zeigt die Namen von „Unix look alikes" mit den jeweiligen Herstellerfirmen. *Bild 3* zeigt die Namen von Unix-Portierungen mit Herstellern und die Version, auf der die Portierung basiert.
Die Auswahl bei beiden Bildern ist willkürlich und erhebt keinen Anspruch auf Vollständigkeit.

Bild 2. Unix „look alike" Systeme

Name	Anbieter
Idris	Whitesmith
Coherent	Mark Williams
Unos	Charles River Data Systems
Regulus	Alcyon Corp.
Primos	Prime Computer Gmbh

Bild 3. Unix-Portierungen

Name	Anbieter	basierend auf
UTS	Amdahl	System V
HP-UX	Hewlett-Packard	System V
XELOS	Perkin Elmer	System V
XENIX	Micro Soft	System V
SINIX	Siemens	XENIX, System III
MUNIX	PCS	System V
ULTRIX32	Digital Equipment	4.2 BSD

Vor- und Nachteile des Systems

Unix läuft auf vielen Computern – vom Mikro bis zum Großrechner. Derzeit gibt es weltweit etwa 400 000 Installationen. Woraus resultiert dieser Erfolg? Die große Akzeptanz liegt sicher darin, daß Unix Implementierungskonzepte enthält, die sowohl sehr leistungsfähig als auch elegant sind. Zu nennen sind das Dateisystem, die Ablauf- und Prozeßsteuerung und die Behandlung der Peripherie (also Drucker, Speichermedien usw.).
Unix ist ein Multiuser- und Multitasking-Betriebssystem. Durch die Mehrbenutzerfähigkeit wird der Prozessor besser ausgenutzt. Die Möglichkeit, mehrere Prozesse quasi gleichzeitig ablaufen zu lassen, erhöht die Effizienz des einzelnen Benutzers.
Die Benutzeroberfläche – die Shell – ist wesentlich leistungsfähiger als man das von anderen Betriebssystemen gewohnt ist. Die Shell ist selbst eine erweiterungsfähige, dialogorientierte Programmiersprache auf höherer Ebene. Mit der Shell ist es möglich, das System allen Anforderungen anzupassen. Des weiteren ist die Shell kein Bestandteil des Kernsystems. Man kann sie durch selbstgeschriebene Menüsysteme ersetzen oder ergänzen.
Unix enthält eine Vielzahl von Werkzeugen, Utilities genannt. Diese können beliebig miteinander kombiniert werden. Bei anderen Systemen müssen diese Werkzeuge erst selbst programmiert werden. Der Einsatz der Werkzeuge entspricht den Grundsätzen des modernen Software-Engineering, deren wichtigste Forderungen Modularität und eine übersichtliche Struktur sind.
In herkömmlichen Betriebssystemen sind Betriebssystemfunktionen ein fester Bestandteil. Bei Unix ist es dagegen weder geheimnisvoll noch schwierig, neue Werkzeuge zu generieren. Genauso

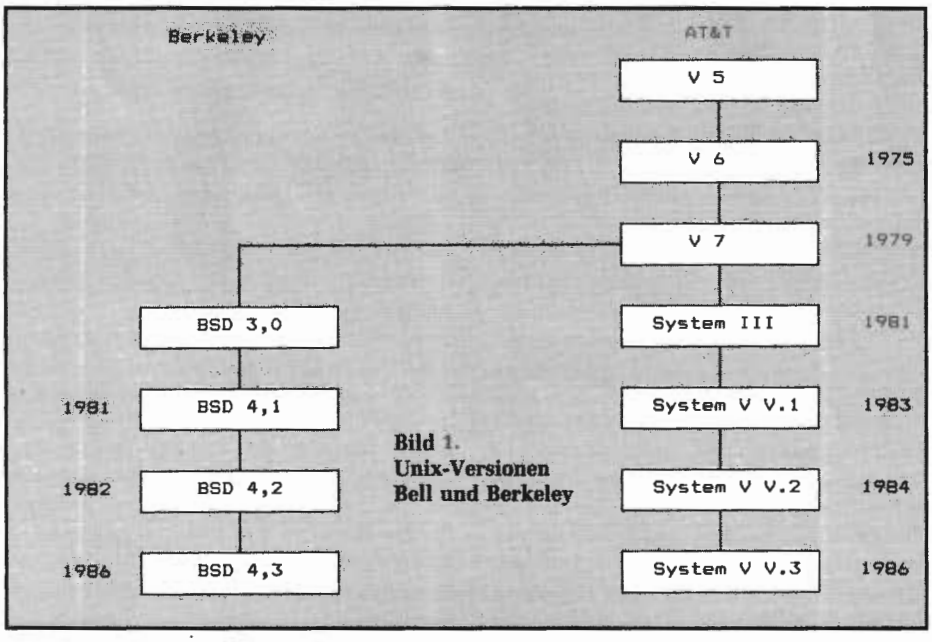


Bild 1. Unix-Versionen Bell und Berkeley

einfach können neue Werkzeuge mit alten verbunden und damit ein neues „Superwerkzeug" geschaffen werden.
Unix ist eine Time-Sharing-Betriebssystem. Jeder Prozeß bekommt den Prozessor für eine bestimmte Zeitscheibe zugeteilt, danach wird der nächste Prozeß bedient. Anwender, die mit solchen Betriebssystemen arbeiten, wollen untereinander Informationen austauschen und gemeinsam auf die Peripherie zugreifen können. Die meisten anderen Betriebssysteme sind dafür nicht ausgelegt. Sie haben Sicherheitseinrichtungen, die es den Benutzern schwer machen, untereinander Informationen auszutauschen. Dagegen bietet Unix eine gute Umgebung für Programmierer, die miteinander am selben Projekt arbeiten.
Auch für Anbieter ist Unix attraktiv. Die Entwicklung neuer Hardware erfolgt in immer kürzeren Abständen. Die Preise der Endprodukte sinken durch die auf den Markt drängende Konkurrenz. Will ein Hersteller dem Kostendruck standhalten, so muß er die Entwicklungskosten so klein wie möglich halten. Unix bietet sich hier als bereits bekanntes und portables Betriebssytem an. Da Unix zu 90 % in C geschrieben ist, benötigt man nur einen für den entsprechenden Computer lauffähigen C-Compiler, um 90 % des Betriebssystems zur Verfügung zu haben. Für den Anwender bedeutet diese Portabilität, daß seine selbstentwikkelten Programme leicht auf andere Unix-Systeme übertragbar sind. Damit ist er auch nicht mehr von einem bestimmten Hersteller abhängig.
Mit Unix ist es zum erstenmal möglich, unterschiedliche Hardware verhältnismäßig einfach zu vernetzen. In Standard-Unix ist ein Netzwerkprotokoll enthalten. Mit diesem Protokoll läßt sich ein homogenes Netz auf einer inhomogenen Hardware verwirklichen. Auch der Anschluß an X-25 (Datex-P) ist mit diesem Protokoll relativ einfach.
Da es kein vollkommenes Betriebssystem gibt, seien auch die Nachteile von Unix nicht verschwiegen.
Der gravierenste ist die Unix-Syntax, die vor allem dem gelegentlichen Benutzer den Angstschweiß auf die Stirn treibt. Die Syntax für die Werkzeuge ist zwar häßlich, aber zumindest durchgängig. Die Namen der Standard-Werkzeuge sind teilweise sinnvolle Abkürzungen. Es gibt allerdings auch den Fall, daß ein Programmname aus den Anfangsbuchstaben der Autoren besteht.
Die Fehlermeldungen, so spärlich sie auch sind, lassen manchmal den Verdacht aufkommen, daß die Autoren der englischen Sprache nicht mächtig waren. Man kann Unix nicht den Vorwurf machen, daß es geschwätzig ist. Fehlermeldungen erscheinen im Höchstfall, wenn eine fehlerhafte Eingabe erkannt wurde. Hat man einen Befehl erfolgreich eingegeben, so wird der Befehl ausgeführt und man erhält kommentarlos die nächste Eingabeaufforderung, nach dem Motto:
„Keine Nachrichten sind gute Nachrichten."
Wird ein Befehl falsch eingegeben, kann es vorkommen, daß das System nichts ausführt und auch keine Meldung ausgibt, denn:
„Wer mag schon schlechte Nachrichten."
Ein für kommerzielle Anwender gravierender Nachteil ist, daß Unix keine Datei-Zugriffsmethoden wie ISAM (Indexed Sequential Access Method) oder ähnliches kennt. Für Unix ist eine Datei ein Bytestrom ohne eine Struktur. Dies hat den Vorteil, daß alle vorhandenen Werkzeuge auf alle Dateien anwendbar sind. Für einen kommerziellen Anwender ist eine Datei, die ein Bytestrom ist, ein Unsinn in sich selbst.
Ein weiteres Manko ist die Systemliteratur. Hier schlägt der Slang der Entwickler voll durch. Es kommt durchaus vor, daß man zwar jedes einzelne Wort versteht, aber der Sinn des Satzes völlig unklar bleibt. Diesen Effekt hat man oft noch verstärkt bei den deutschen Handbüchern einiger Hersteller.

Unix für wen

Unix ist ein Betriebssystem, das ursprünglich für Programmentwickler geschrieben wurde. Deswegen bietet es zur Programmentwicklung besonders viele Werkzeuge. Andererseits ist es einfacher, eine Applikation auf dem Zielsystem für das Zielsystem zu schreiben. Mittlerweile gibt es zahlreiche Unix-Applikationen für alle möglichen Anwendungsgebiete.
Bei Turnkey-Systemen sieht der Anwender das darunterliegende Betriebssystem überhaupt nicht. Wenn er das System startet, landet er sofort in seiner Anwendung. Die Anwendung beinhaltet auch notwendige Funktionen wie Datensicherung und sonstige Systemverwaltungstätigkeiten.
Viele Anwender wollen zwar bestimmte Programmpakete fertig übernehmen, sich jedoch den Weg für eigene Software freihalten. Für diesen Fall stellen die Hersteller oft andere Oberflächen als die Shell zur Verfügung.
Entscheidet man sich für ein Turnkey-System, so sollte man darauf achten, daß neben der Anwendung auch Betriebssystemfunktionen enthalten sind, so daß man nicht mehr gezwungen ist, sich mit dem Betriebssystem an sich herumzuschlagen. Auch wenn der Anwender von solchen Systemen sich für das darunterliegende Betriebssystem nicht mehr interessiert, so ist es dennoch für ihn wichtig, daß seine Anwendung auch dann sehr leicht und kostengünstig auf einen anderen Unix-Rechner übertragen werden kann. Er ist nicht mehr an einen bestimmten Hardware-Hersteller gebunden.

Vor- und Nachteile von Menüsystemen

Dem ungeübten Benutzer kommt die Bedienerführung entgegen, jedoch wer sich zum x-ten Mal durch das siebte Untermenü hangelt, hätte lieber einen schnelleren Weg zur Verfügung, um dasselbe zu erreichen.
Standard-Unix ist so umfangreich, daß kein Menüsystem in der Lage ist, alle Funktionen abzudecken. Vor allem deswegen nicht, da es unter der Shell möglich ist, vorhandene Funktionen so miteinander zu verbinden, daß das Ergebnis eine ganz neue Funktion ergibt.

Der normale Anwender nutzt höchstens 20 % der insgesamt zur Verfügung stehenden Werkzeuge. Abhängig von der jeweiligen Anwendung werden jedoch andere 20 % des Funktionsumfangs genutzt. So ist es für den Programmierer, der ein Menüsystem entwirft, schwierig zu entscheiden, was er dem Anwender zur Verfügung stellt. In jedem vernünftigen Menüsystem sollte es deshalb möglich sein, das Menüsystem zu verlassen und mit der normalen Shell zu arbeiten.

Systemkomponenten

Bis hierher wurde immer von dem Unix-System an sich gesprochen. Da Unix ein modulares System ist, besteht es aus einzelnen Komponenten. Im folgenden werden sie nur aufgezählt. Später sollen diese Komponenten näher beschrieben werden.
Der Teil, der direkt auf der Hardware aufsetzt, ist der sogenannte Kernel. Der Kernel verteilt die verfügbaren Ressourcen auf die einzelnen Benutzer, von denen die wichtigste die CPU-Zeit ist.
Ein weiterer Teil ist das Dateisystem. Durch seine hierarchische Struktur ist es für den Benutzer einfach, die Logik seiner Anwendung darauf abzubilden.
Das wichtigste Handwerkzeug für den Benutzer ist die Kommandosprache. Es

ist die Shell, die aber in ihrem Funktionsumfang weit über übliche Kommandosprachen hinausgeht.
Zu jedem Unix-System gehören die vielfältigen Werkzeuge (Tools), die den Anwender davon entbinden, sich selbst diese Hilfsmittel zu schaffen, oder gar teuer zu kaufen.
Ein weiterer Punkt sind die Netzwerke, von denen mindestens ein Netzwerkprotokoll standardmäßig enthalten ist.

Der Systemkern

Der Systemkern, meist Kernel genannt, ist der Teil des Betriebssystems, den der Benutzer am wenigsten sieht. Er ist als eine Schale um die Hardware zu betrachten. Normalerweise erfolgt die Interaktion des Benutzers nicht mit dem Kernel, sondern über Hilfsprogramme (z. B. die Shell), die ihrerseits über sogenannte System-Calls mit dem Kernel korrespondieren.
Obwohl der Kernel für den Benutzer wenig sichtbar ist, stellt er ihm Dienste zur Verfügung, ohne die er das System nicht erfolgreich nutzen könnte.
Der Kernel erlaubt einem Prozeß die Nutzung des Prozessors. Ist die Zeitscheibe abgelaufen oder der Prozeß nicht mehr lauffähig, weil er auf ein äußeres Ereignis warten muß (z. B. auf Daten von der Peripherie), so teilt der Kernel den nächsten lauffähigen Prozeß dem Prozessor zu.
Einem laufenden Prozeß wird vom Kernel Speicherplatz zugeteilt. Normalerweise ist in einem System nicht genügend Hauptspeicher für alle lauffähigen Prozesse vorhanden. Der Kernel sorgt dafür, daß sich die vom Prozessor benötigte Daten im Hauptspeicher befinden. Desweiteren lagert er die Daten von momentan nicht lauffähigen Prozessen auf Sekundärspeicher (z. B. eine Festplatte) aus.
Der Kernel verwaltet den Sekundärspeicher und etabliert damit das Dateisystem. Er sorgt dafür, daß Benutzerdaten gespeichert und auch wiederverwendet werden können. Daten werden vor nicht erlaubtem Zugriff geschützt.
Für die Interaktion mit der Peripherie ist ebenfalls der Kernel verantwortlich. In der Syntax besteht kein Unterschied, ob der Benutzer mit einer Datei oder z. B. mit einem Gerät wie einem Terminal oder Drucker arbeitet. Der Kernel erkennt, auf welches Gerät die Ausgabe erfolgt und aktiviert den entsprechenden Treiber.
Eines der wesentlichen Kennzeichen des Unix-Systems ist, daß der eigentliche Kern des Systems nur sehr klein ist, aber eine wohldefinierte Schnittstelle nach außen besitzt.
Viele Funktionen, die in anderen Betriebssystemen unveränderbare Bestandteile des Kerns sind, wurden hier in die Werkzeuge verlagert. Damit sind sie anpaß- oder austauschbar. Der Kernel stellt sich für den Unix-Anwender als eine Menge von Aufrufen dar, die er entweder direkt, oder über den Umweg der Shell und der vorhandenen Werkzeuge benutzen kann, die dann ihrerseits diese Aufrufe verwenden.

Dateisystem mit Hierarchie

Eine Datei ist eine zusammengehörige Informationsmenge, die unter einem bestimmten Namen auf einem Massenspeicher abgelegt wird. Das Dateisystem ist der organisatorische Rahmen für eine Gruppe von Dateien. Bei vielen Betriebssystemen besteht die Organisation von Dateien darin, daß sie einen großen unsortierten Haufen bilden. Solange das System kein Mehrbenutzersystem ist, kann das auch durchaus genügen. Bei Mehrbenutzersystemen, wo die Anzahl der Dateien von mehreren Hundert bis zu vielen Tausenden variiert, ist eine solche Art der Organisation nicht geeignet.
Im Unix-System werden Dateien, die logisch zusammengehören, in Dateiverzeichnissen – sogenannten Directories – zusammengefaßt. Diese Art einer einstufigen Organisation kennen auch einige andere Betriebssysteme. Unix geht hier noch weiter: Die Anzahl der Ebenen ist beliebig und nur durch die Größe des Massenspeichers beschränkt. Jedem Benutzer ist ein „Home-Directory" zugeordnet, in dem er beliebig weitere Dateiverzeichnisse erstellen kann. Man sagt auch das Unix-Dateisystem ist hierarchisch. Das oberste Dateiverzeichnis ist das sogenannte Wurzel-Verzeichnis, auch Root-Directory genannt. Das Zeichen für dieses Directory ist „/". Im Root-Directory stehen weitere Dateiverzeichnisse, in denen wiederum Dateiverzeichnisse enthalten sein können. Die einzelnen Namen werden durch „/" voneinander getrennt. Wird der Name einer Datei von Root aus angegeben, so sagt man dazu auch Pfad. Der Pfad einer Datei könnte z. B. „/usr/hugo/zahl.c" lauten. Dies bedeutet, daß unter Root ein Directory existiert, das „usr" heißt. Unter „usr" gibt es ein weiteres Directory mit dem Namen „hugo". In dem Directory hugo gibt es eine Datei, deren Name „zahl.c" ist. Ein typisches Unix-Dateisystem zeigt *Bild 4*.
Unter Unix können in einem System durchaus zwei Dateien mit demselben Namen existieren, solange sie sich durch ihren Pfadnamen unterscheiden. Unix kennt drei unterschiedliche Dateiarten:
Normale Dateien
Dateiverzeichnisse (Directories)
Gerätedateien (Special Files)

Normale Dateien

Eine normale Datei wird benutzt, um Informationen zu speichern. Eine Datei kann z. B. ein ausführbares Programm enthalten, den Text eines Dokumentes oder alle Personaldaten einer Firma. Bis zu 14 Zeichen darf der Dateiname lang sein. Zwei Dateien im selben Dateiverzeichnis können nicht denselben Namen haben, da der Pfad derselbe ist. Der Name sollte keine nichtdruckbaren Zeichen enthalten und auch keine Shell-Metazeichen (wird in einer späteren Folge erklärt). Ansonsten muß man bei der Namensgebung keine Konventionen beachten. Eine Ausnahme von dieser Regel bilden Quelldateien. Die Compiler erwarten, daß bestimmte Konventionen eingehalten werden. So müssen z. B. C-Quelldateien mit ".c" aufhören, damit sie der C-Compiler bearbeiten kann.
Es gibt zwei Arten von Normal-Dateien, Textdateien und Binärdateien. Textdateien werden als 7-Bit ASCII-Code abgespeichert. Binärdateien können in jedem Byte einen der 256 möglichen Werte enthalten. Binärdateien sind normalerweise ausführbare Programme, die dadurch entstanden sind, daß ein Quellprogramm übersetzt und geladen wurde. Abgesehen von diesem Unterschied impliziert und kennt Unix keine wie auch immer geartete Struktur. Eine Datei ist ein Bytestrom und nichts weiter. Diese nicht vorhandene Struktur ist einer der Gründe, warum jedes Dienstprogramm mit jeder Datei arbeiten kann.

Dateiverzeichnisse

Ein Dateiverzeichnis ist eine Liste von Dateien oder weiterer Dateiverzeichnisse. In einem Dateiverzeichnis wird einer Datei eine Nummer, die sogenannte I-Node zugeordnet. Diese I-Node ist ein Zeiger in eine Index-Liste, in der weitere Informationen über die Datei enthalten sind. Diese weiteren Informationen beinhalten unter anderem die Zugriffsrechte, den Namen des Eigentümers, Datum und Uhrzeit der Erstellung und der letzten Modifikation sowie die Nummern der Blöcke, in denen der Dateiinhalt zu finden ist.
Jedem Benutzer wird vom System-Administrator sein Home-Directory zu-

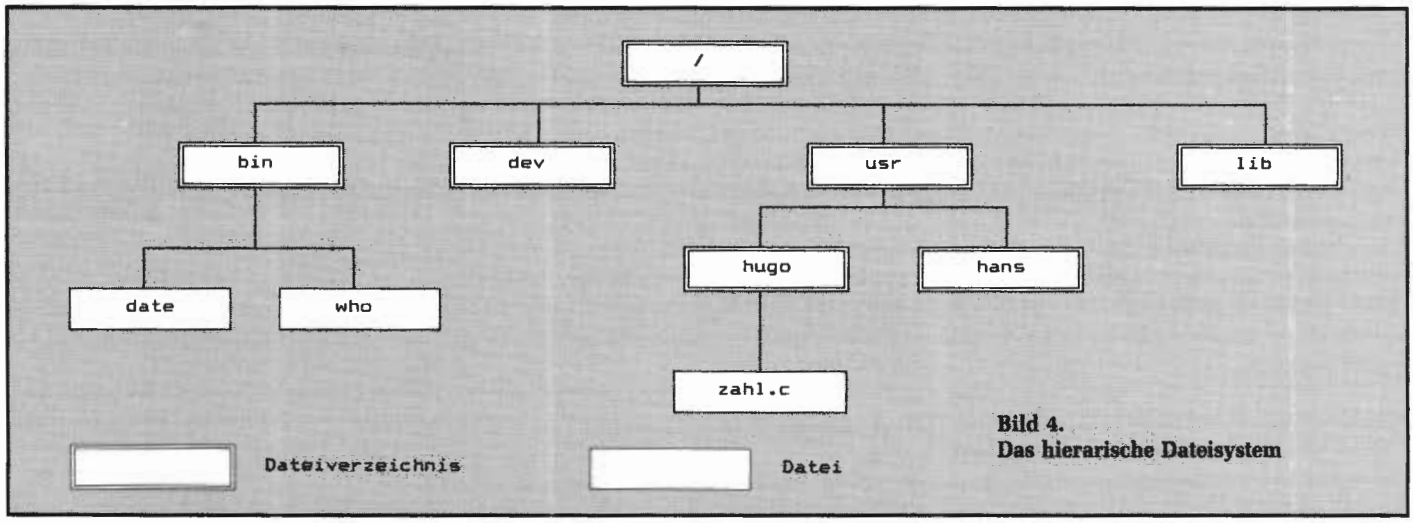


Bild 4.
Das hierarische Dateisystem

gewiesen. In diesem Directory befindet sich der Benutzer, nachdem er eine gültige Benutzerkennung eingegeben hat. Natürlich kann er in ein anderes Dateiverzeichnis wechseln. Das System kennt immer das momentane Directory eines Benutzers. Jedes Directory außer der Root hat ein Parent-Directory (Eltern-Verzeichnis). Es gibt für das momentane und das Parent-Directory Synonyme. Das Synonym für das momentane Directory ist ".", das Synonym für das Parent-Directory "..". Hat ein Benutzer ein Home-Directory, das „/usr/hugo" heißt und hat er sein Dateiverzeichnis noch nicht gewechselt, so entspricht "." dem Namen „/usr/hugo" und ".." dem Namen „/usr". „/usr" ist also Parent-Directory von „hugo", „hugo" ist Child (Kind) von „/usr", „/" ist Parent von „usr". Die Synonyme "." und ".." tragen zwar kräftig zur Verwirrung bei, ersparen aber dem Anwender einiges an Schreibarbeit.

Gerätedateien (Special Files)

Der grundlegende Unterschied zwischen Geräte- und Normaldateien besteht darin, daß Gerätedateien keine Informationen beinhalten. Der Zweck einer Gerätedatei besteht darin, dem Benutzer einen Ein/Ausgabe-Kanal zur Verfügung zu stellen.
Gerätedateien werden per Konvention in dem Directory „/dev" angelegt. Listet sich der Anwender den Inhalt von „/dev" aus, so kann er daraus ersehen, welche Peripherie an den Rechner angeschlossen ist. Jedes an den Rechner angeschlossene Terminal, jeder Drucker, jeder Plotter, jedes Magnetband, jede Platte hat hier einen oder mehrere Einträge.
Ein Eintrag in „/dev" besteht unter anderem aus dem logischen Gerätenamen, und aus der sogenannten „major" und „minor" Nummer. Anhand der „major" Nummer erkennt das System den entsprechenden Treiber, der den Datentransfer mit dem E/A-Kanal in der richtigen Weise aufsetzt und durchführt. Die „minor" Nummer enthält die physikalische Adresse des entsprechenden Gerätes.
Unter Unix ist es sehr einfach, dem Rechner ein zusätzliches Gerät bekanntzumachen. Im Prinzip benötigt man dazu einen Befehl und nicht, wie bei manchen Systemen, eine stundenlange Neugenerierung des Betriebssystems. Durch dieses Konzept sieht der Benutzer keinen Unterschied, ob er mit einem Gerät oder mit einer Datei arbeitet. Auch die Anwendungsprogramme „wissen" nicht, von wo sie lesen oder wohin sie schreiben. Für den Benutzer ist dadurch das Arbeiten mit dem System erheblich einfacher. Beim Schreiben von Anwendungsprogrammen, das E/A-Operationen durchführen, braucht sich der Programmierer nicht um gerätespezifische Feinheiten zu kümmern.

Zugriffsberechtigungen

Da Unix ein Mehrbenutzer-System ist, muß es möglich sein, die Dateien eines Benutzers vor unberechtigtem Zugriff eines anderen Benutzers zu schützen. Für jede Datei gibt es eine aus neun Bit bestehende Kennung, um die verschiedene Zugriffsmöglichkeiten zu markieren. Ein Benutzer ist vom Dateisystem her gesehen entweder der Eigentümer einer Datei, ein Angehöriger der Gruppe des Eigentümers oder er gehört zum Rest der Welt. Für jeden dieser drei Benutzertypen gibt es wieder drei verschiedene Zugriffsberechtigungen, nämlich Lese-, Schreib- und Ausführberechtigung.
Was die Lese- und Schreibberechtigung bei einer Datei bedeutet, muß nicht näher erläutert werden. Die Ausführberechtigung gibt an, daß es sich um ein ausführbares Programm handelt. Ein ausführbares Programm kann durch Compilieren eines Quellprogrammes entstehen oder eine Shellprozedur sein. Der jeweilige Benutzertyp hat das Recht, das entsprechende Programm zu starten. Bei einem Dateiverzeichnis bedeutet Lesen, daß der jeweilige Benutzertyp sehen darf, welche Dateien oder Dateiverzeichnisse das Directory beinhaltet. Schreiben bedeutet, daß er hat das Recht hat, eine Datei in diesem Directory zu kreieren. Die Ausführberechtigung bedeutet, daß er durch dieses Directory durchgreifen darf und damit sein momentanes Directory in ein Unterdirectory dieses Verzeichnisses ändern kann. Alle neun Berechtigungsbit können in beliebiger Kombination gesetzt oder gelöscht werden. Darf eine bestimmte Datei nicht gelöscht werden, so genügt es nicht, die Schreibberechtigung für die Datei wegzunehmen, sondern auch die Schreibberechtigung im Directory muß entfernt werden. Das Löschen einer Datei verändert nicht die Datei sondern nur das Dateiverzeichnis. Hat man Schreiberlaubnis für das Directory kann man also beliebig Dateien löschen. Das ist typische Unix-Logik.

Literatur:

[1] *Thomas, Rebecca Anne* und *Yates, Jean:* A userfriendly introduction to the Unix operating system. Computerworld, Februar 1982, Seite 19.
[2] *Norman, Donald A.:* The truth about Unix. Arbeitspapier des Department of Psychology and Programm in Cognitive Science. University of California, San Diego. April 1981.

H. Herre und H. Völz

GOTO und Standard-Basic

Im letzten Jahrzehnt hat die strukturierte Programmierung beträchtlich an Bedeutung gewonnen. Erzwingt sie doch eine klarere Übersicht über die Programme und führt unter anderem zu fehlerarmen Programmen. Der GOTO-Befehl ist dabei in Verruf geraten, denn seine großzügige Anwendung kann schnell zum sogenannten GOTO-Salat führen. Hinzu kommt, daß in fast allen Basic-Dialekten der GOTO-Befehl beträchtliche Zeit verbraucht und damit zu langsamen Programmen führt. Andererseits läßt sich zeigen, daß der GOTO-Befehl unverzichtbar ist. Auf diese Problematik soll dieser Beitrag aufmerksam machen.

Es sei erwähnt, daß die allgemeine WHILE-DO-Befehlsfolge das GOTO selbst im allgemeinen Fall zu vermeiden erlaubt. Sie ist jedoch nur in großen Basic-Dialekten enthalten.

Einfache Loop-Programme

Ein einfaches Loop-Programm soll neben der Standard-Arithmetik (die vier Grundrechnungsarten) nur die Befehle

FOR TO NEXT

und

IF THEN

enthalten. Aus der mathematischen Logik läßt sich ableiten, daß mit so aufgebauten Programmen alle primitiv-rekursiven Funktionen berechenbar sind. Ferner ist bekannt, daß ein einfaches Loop-Programm für alle Zahlen zu einem Ende führt. Es wird also ein Ergebnis erreicht oder das Programm bricht wegen Zahlenüberlauf ab; in der Fachsprache bedeutet dies, daß ein solches Programm stets terminiert.

Die allgemein-rekursiven Funktionen, die auf jeder Stufe noch ein Suchproblem enthalten, sind nicht mit einfachen Loop-Programmen berechenbar. Hierzu zählt zum Beispiel die Ackermannsche Funktion, auf die in einem späteren Beitrag eingegangen wird.

Es gibt aber auch berechenbare Funktionen, von denen zur Zeit nicht bekannt ist, ob sie primitiv rekursiv sind oder ob sie für alle Argumente definiert sind, d. h. ob das berechnende Programm für alle Eingaben terminiert. Ein Beispiel liefert das folgende Problem. Bei ihm ist damit die Frage offen, ob es ohne GOTO programmierbar ist oder nicht. In gewissem Sinne also auch eine Aufgabenstellung für Knobler.

```
10 REM:Problem Hydra; Version 1
20 INPUT"Zahl = ";A: A=INT(A)
30 IF A=1 THEN END
40 B=A/2:IF B =INT(B) THEN A=B: GOTO 60
50 A=3*A+1
60 PRINT A,:GOTO 30
READY

RUN

Zahl = ? 167
 502          251          754          377
 1132         566          283          850
 425          1276         638          319
 958          479          1438         719
 2158         1079         3238         1619
 4858         2429         7288         3644
 1822         911          2734         1367
 4102         2051         6154         3077
 9232         4616         2308         1154
 577          1732         866          433
 1300         650          325          976
 488          244          122          61
 184          92           46           23
 70           35           106          53
 160          80           40           20
 10           5            16           8
 4            2            1
READY
```

Bild 1. Hydra – gar nicht harmlos

```
10 REM: Programm fuer die Hydrafunktion
40 Z=999999 : I=6
50 FOR Y=1 TO Z: A=Y
60 : FOR X=1 TO Z
70 : B=A/2: IF B=INT(B) THEN A=B : GOTO 90
80 : A=3*A+1
90 : IF A=1 THEN I=X: X=Z: PRINT"F("Y;") =";I
100 NEXT X: NEXT Y
READY

RUN
F( 1 ) = 3
F( 2 ) = 1
F( 3 ) = 7
F( 4 ) = 2
F( 5 ) = 5
F( 6 ) = 8
F( 7 ) = 16
F( 8 ) = 3
F( 9 ) = 19
F( 10 ) = 6
F( 11 ) = 14
F( 12 ) = 9
F( 13 ) = 9
F( 14 ) = 17
F( 15 ) = 17
F( 16 ) = 4
F( 17 ) = 12
F( 18 ) = 20
F( 19 ) = 20
F( 20 ) = 7
F( 21 ) = 7
F( 22 ) = 15
F( 23 ) = 15
F( 24 ) = 10
F( 25 ) = 23
F( 26 ) = 10
F( 27 ) = 111
F( 28 ) = 18
F( 29 ) = 18
F( 30 ) = 18
F( 31 ) = 106
F( 32 ) = 5
F( 33 ) = 26
F( 34 ) = 13
F( 35 ) = 13
F( 36 ) = 21
F( 37 ) = 21
F( 38 ) = 21
F( 39 ) = 34
F( 40 ) = 8
F( 41 ) = 109
F( 42 ) = 8
F( 43 ) = 29
F( 44 ) = 16
F( 45 ) = 16
F( 46 ) = 16
F( 47 ) = 104
F( 48 ) = 11
F( 49 ) = 24
F( 50 ) = 24
F( 51 ) = 24
F( 52 ) = 11
F( 53 ) = 11
F( 54 ) = 112
F( 55 ) = 112
F( 56 ) = 19
F( 57 ) = 32
F( 58 ) = 19
F( 59 ) = 32
F( 60 ) = 19
F( 61 ) = 19
F( 62 ) = 107
F( 63 ) = 107
F( 64 ) = 6
F( 65 ) = 27
F( 66 ) = 27
F( 67 ) = 27
F( 68 ) = 14
F( 69 ) = 14
F( 70 ) = 14
F( 71 ) = 102
F( 72 ) = 22
F( 73 ) = 115
F( 74 ) = 22
F( 75 ) = 14
F( 76 ) = 22
F( 77 ) = 22
F( 78 ) = 35
F( 79 ) = 35
F( 80 ) = 9
F( 81 ) = 22
F( 82 ) = 110
F( 83 ) = 110
F( 84 ) = 9
F( 85 ) = 9
F( 86 ) = 30
F( 87 ) = 30
F( 88 ) = 17
F( 89 ) = 30
F( 90 ) = 17
F( 91 ) = 92
F( 92 ) = 17
F( 93 ) = 17
F( 94 ) = 105
F( 95 ) = 105
F( 96 ) = 12
F( 97 ) = 118
F( 98 ) = 25
F( 99 ) = 25
F( 100 ) = 25
F( 101 ) = 25
F( 102 ) = 25
F( 103 ) = 87
F( 104 ) = 12
F( 105 ) = 38
F( 106 ) = 12
F( 107 ) = 100
F( 108 ) = 113
F( 109 ) = 113
F( 110 ) = 113
F( 111 ) = 69
F( 112 ) = 20
F( 113 ) = 12
F( 114 ) = 33
F( 115 ) = 33
F( 116 ) = 20
F( 117 ) = 20
F( 118 ) = 33
F( 119 ) = 33
F( 120 ) = 20
F( 121 ) = 95
F( 122 ) = 20
F( 123 ) = 46
F( 124 ) = 108
F( 125 ) = 108
F( 126 ) = 108
F( 127 ) = 46
F( 128 ) = 7
F( 129 ) = 121
F( 130 ) = 28
F( 131 ) = 28
F( 132 ) = 28
F( 133 ) = 28
F( 134 ) = 28
```

Bild 2. Das Programm Hydrafunktion

```
10 REM:Programm zur Bestimmung jener Eingaben,
20 REM:die mit einer immer groesseren Anzahl
30 REM:von Durchlaeufen zur 1 fuehren
40 INPUT"Zahl der Durchlaeufe ="; Z
50 FOR Y=9 TO 999999: A=Y
60 : FOR X=1 TO Z
70 : B=A/2: IF B=INT(B) THEN A=B : GOTO 90
80 : A=3*A+1
90 : IF A=1 THEN X=Z: NEXT X: NEXT Y: END: PRINT Y;":";I,
100 NEXT X: PRINT Y,: NEXT Y
READY
RUN
Zahl der Durchlaeufe = ? 100
 27          31          41          47
 54          55          62          63
 71          73          82          83
 94          95          97          108
 109         110         124         125
 126         129         142         143
 145         146         147         164
 165         166         171         188
 189         190         193         194
 195         199         214         215
 216         218         220         221
 231         235         248         250
 252         253         257         258
 259
BREAK IN 80
READY
RUN
Zahl der Durchlaeufe = ? 150
 703         763         775         871
 937         1017        1033        1055
 1071        1161        1215        1249
 1263        1307        1351        1377
 1406        1407        1471        1519
 1526        1527        1537        1550
 1551        1563        1583        1607
 1665        1695        1742        1743
 1801        1819        1823        1839
 1874        1875        1895        1915
 1951        1961        2023        2025
 2027        2034        2035        2043
 2047        2049        2066        2067
 2107        2110        2111        2142
 2143        2155        2207        2223
 2279        2281        2290        2291
 2299        2306        2307        2311
 2322        2323        2326        2327
 2329
BREAK IN 70
READY
```

Bild 5. Untersuchung der Zahl der Durchläufe

Das Problem Hydra

Es wird mit irgendeiner natürlichen Zahl begonnen. Sie kann z. B. mit INPUT eingegeben werden. Danach wird in die Prüfschleife eingetreten.

- ☐ Ist die Zahl geradzahlig, so wird sie halbiert.
- ☐ Ist sie ungeradzahlig, so wird sie mit 3 multipliziert und noch um 1 erhöht.

Danach folgt wieder Eintritt in die Prüfschleife. Das Programm bricht nur dann ab, wenn in der Prüfschleife die Zahl 1 erreicht wird. Ob dieser Fall für jede eingegebene Zahl auftritt, ist nach unseren Kenntnissen bis heute nicht geklärt.

```
10 REM: Problem Hydra; Version 2
20 INPUT"Zahl =."; A: A=INT(A)
30 FOR X=1 TO 999999
40 PRINT A,: IF A=1 THEN END
50 B=A/2: IF B =INT(B)
   THEN A=B: NEXT X: GOTO 70
60 A=3*A+1: NEXT X
70 PRINT A,: GOTO 40
80 GOTO 30
90 PRINT"Programm ist noch nicht
   zu Ende"
READY
```

Bild 3. Das Programm aus Bild 2, zur Änderung der Laufparameter ergänzt

```
10 REM: Problem Hydra; Version 3
20 INPUT"Zahl = ";A: A=INT(A)
30 FOR X=1 TO 2 STEP 0: PRINT A,
40 IF A=1 THEN END
50 B=A/2: IF B=INT(B) THEN A=B: NEXT X
60 A=3*A+1: NEXT X
READY
```

Bild 4. Hydra, mit Step 0 besonders gefährlich

Das Problem läßt sich wie folgt veranschaulichen. Es existiere eine Hydra oder ein Drachen mit der Fähigkeit, Köpfe nachzubilden. Immer, wenn er eine gerade Zahl von Köpfen hat, können ihm die Hälfte der Köpfe abgeschlagen werden. Die Hydra ist zu töten, wenn sie nur noch einen Kopf hat.

Ein mögliches Basic-Programm P1 für die Formalisierung des genannten Verfahrens zeigt *Bild 1*. Den Lauf des Programms P1 für die Zahl 167 zeigt Bild 1 ebenfalls. Auffällig ist das „unregelmäßige" Zu- und Abnehmen der „Köpfe". Mit P1 läßt sich die folgende Funktion f(x) definieren:

$$f(x) = \begin{cases} \text{Anzahl der Durchläufe bei Eingabe von x, falls P1 bei Eingabe von x terminiert;} \\ \text{nicht definiert, andernfalls.} \end{cases}$$

Die Funktion f(x) läßt sich durch das Programm P2 (*Bild 2*) berechnen. Nach dem oben Gesagten ist jedoch nicht bekannt, ob f(x) primitiv rekursiv ist, also ob sich f(x) durch ein einfaches Loop-Programm programmieren läßt. Deshalb werde das Programm auf einen endlichen Bereich (hier 999999) begrenzt; auf diese Weise läßt sich das zeitaufwendige GOTO vermeiden. Dafür kann aber auch nur ein Teil der Werte von f(x) berechnet werden.

Die ersten Werte der Funktion sind in Bild 2 gezeigt. Die Unregelmäßigkeit der Funktion kommt deutlich zum Ausdruck. Auffallend ist das häufige Auftreten von „Zwillingen", d. h. von aufeinanderfolgenden Zahlen, für die die Funktion den gleichen Wert hat.

Möglichkeit einer experimentellen Entscheidung

Das Problem besteht darin, ob die vollständige Funktion f ohne GOTO-Befehle programmiert werden kann. Hierzu wäre als erstes zu klären, ob das Programm P1 für alle Eingaben terminiert. Da dies nicht bekannt ist, könnte man die Terminierung experimentell zu entscheiden versuchen. In jedem Basic-Dialekt existiert eine größte Integerzahl; sie liegt z. B. bei einer sechsstelligen Mantisse bei 999999. Diese Grenze der ganzen Zahlen stellt zugleich die experimentelle Schranke des Nachweises der Terminierung dar. Es läßt sich nun die Frage stellen, ob sich wenigstens in dem durch den Basic-Dialekt festgelegten Bereich der ganzen Zahlen die Terminierung bzw. Nicht-Terminierung experimentell entscheiden ließe. Die Terminierung ist in diesem Bereich dann gegegeben, wenn für alle Zahlen bis zum zulässigen Maximum bei P1 der Wert 1 als Ende der Schleife nach einer gewissen endlichen Zeit, sagen wir beispielsweise einer Stunde, erreicht ist. Was geschieht jedoch, wenn für eine Eingabe nach einer Stunde oder auch nach 3, 4 bzw. 5 Stunden das Ende nicht erreicht wird? Damit

```
LIST

10 REM: Programm zur Bestimmung jener Eingaben,
20 REM: die mit einer immer groesseren Anzahl
30 REM: von Durchlaeufen zur 1 fuehren
40 Z=999999 : I=6
50 FOR Y=9 TO Z: A=Y
60 : FOR X=1 TO Z
70 :  B=A/2: IF B=INT(B) THEN A=B : GOTO 90
80 :  A=3*A+1
90 :  IF A=1 THEN B=X: X=Z: IF B I THEN I=B: PRINT Y;":";I,
100 NEXT X: NEXT Y
READY

RUN
 9 : 19        18 : 20       25 : 23       27 : 111
 54 : 112      73 : 115      97 : 118      129 : 121
 171 : 124     231 : 127     313 : 130     327 : 143
 649 : 144     703 : 170     871 : 178     1161 : 181
```

Bild 6. Das Verhalten: unregelmäßig

ist immer noch nicht bewiesen, daß das Ende für diese Eingabe nie erreicht wird. Der Beweis, daß das Programm selbst in diesem Zahlenbereich terminiert oder ob es für eine Eingabe nicht terminiert, ist so also nicht zu erbringen. Aus diesem Grunde ist der noch ausstehende theoretische Beweis für eine der beiden Varianten notwendig.
Angenommen wir hätten nun die Terminierung bis 999999 bewiesen, warum sollte es dann nicht möglich sein, mit der Gleitkomma-Arithmetik bis ca. 10^{30} weiter zu rechnen? Dies geht nicht wegen der Rundungsfehler, denn die Addition von 1 ist in diesem Bereich nicht mehr unbeschränkt möglich. Die Zahlen der Gleitkomma-Arithmetik liegen nämlich oberhalb der Integergrenze nicht hinreichend dicht. Als Trost sei aber gesagt, daß zumindest in Basic alle Rechnungen solange dauern, daß es Ihnen kaum möglich sein dürfte, den Test bis 999999 durchzuführen.

Anwendung von FOR-NEXT

Viele GOTO-Sprünge lassen sich in Basic mit FOR NEXT umgehen. So ist bereits das Programm P2 entstanden. Es ist dabei offensichtlich, daß die begrenzte FOR-NEXT-Schleife nicht in jedem Fall zum Erfolg führt, denn der Laufparameter kann eben nur so oft geändert werden, wie es der größten darstellbaren ganzen Zahl entspricht. Deshalb müßten die Zeilen 70 und 80 dem Programm hinzugefügt werden und das eigentliche Programm zeigt dann *Bild 3*.

Kleiner Trick in der FOR-NEXT-Schleife

Die einfache, bisher verwendete FOR-NEXT-Schleife hat die Schrittweite 1. Wie sich die FOR-NEXT-Schleife verhält, wenn nun 1 nicht mehr zu addieren ist (also an der Grenze der größten ganzen Zahl), hängt vom jeweiligen Dialekt ab. Damit dieses Verhalten aber eindeutig wird, gelingt in den meisten Dialekten die Programmierung der Schrittweite 0. Damit entsteht ein Programm (*Bild 4*), welches genau innerhalb der ganzen Zahlen den Test auf Abbruch des Hydraproblems gestattet. Mit ihm lassen sich die Laufzeiten für alle Eingaben im zulässigen Zahlenbereich bestimmen. Hiermit wird praktisch WHILE DO indirekt realisiert.

Bemerkungen

Die Bilder 1 und 2 zeigen ein eigenwilliges Verhalten der Funktion. Es ist, soweit uns bekannt ist, zahlentheoretisch nicht untersucht. Deshalb prüften wir mit weiteren Methoden das Verhalten der Hydrafunktion. Wir ließen uns mit dem Programm P5 alle Eingaben ausdrucken, deren Ausgabewert unterhalb bestimmter Zahlen liegen. Für 100 und 150 zeigt das Ergebnis *Bild 5*. Für 200 konnte nach einer Stunde kein Wert gefunden werden. Deshalb wurde das Programm in *Bild 6* geschrieben, um die Eingaben für ständig steigende Werte zu erhalten. Hier wird wieder die große Unregelmäßigkeit der Ergebnisse sichtbar.
Wir haben den Eindruck, daß Hydra ein lohnendes Objekt für Experimente ist, wollen dabei aber nicht verhehlen, daß eine gute Analyse selbst in einem relativ kleinen endlichen Bereich sehr zeitaufwendig ist. Wahrscheinlich wäre ein Maschinenprogramm notwendig.

Lieber Leser, versuchen Sie Ihr Heil. Vielleicht teilen Sie der Redaktion einige Laufzeiten für relevante Zahlen auf Ihrem Rechner mit.

Siemens PC-D mit hoher Auflösung

In mc 4/87 wurde versehentlich ein falsches Listing abgedruckt. Der Siemens PC-D und der Tandy 2000 sind weder voll IBM-kompatibel, noch untereinander gleich. Die Startadresse des Video-RAMs ist beim PC-D F000H und beim Tandy 2000 E000H. Beim PC-D beträgt die Auflösung 640 × 350 Punkte und beim Tandy 2000 640 × 400 Punkte. Die Assemblerroutinen zum Ansprechen der Grafik sind auf dem Siemens PC-D direkt lauffähig. Um das Programm an den Tandy 2000 anpassen zu können, sind die entsprechenden Stellen markiert.
Turbo-Pascal kann diese externen Assembler-Routinen aufrufen. Die Programmzeilen in Turbo-Pascal lauten:

```
Procedure Plot(x,y); external
'mcgraph.com';
Procedure Radi(x,y); external plot[3];
Function Test(x,y):boolean; external
plot[6];
```

Manfred Weis

```
                        ;*****************************************
                        ;*                                       *
                        ;*    MCGRAPH Routinen zur grafischen    *
                        ;*            Ansteuerung aus TURBO      *
                        ;*                                       *
                        ;*    PLOT   : Punkt setzen              *
                        ;*    RADI   : Punkt loeschen            *
                        ;*    TEST   : Punkt testen              *
                        ;*                                       *
                        ;*****************************************
0000                    code    segment para public 'code'
                                assume  cs:code
0000  EB 07 90                  jmp     plot    ;Punkt setzen
0003  EB 5C 90                  jmp     radi    ;Punkt loeschen
0006  E9 00B1 R                 jmp     test    ;Punkt testen
```

Mit diesem Assembler-Programm wird die hochauflösende Grafik angesprochen

```
0009  55          plot:   push    bp
000A  8B EC               mov     bp,sp   ;Stapelzeiger setzen
000C  53                  push    bx      ;Registerdaten sichern
000D  51                  push    cx
000E  52                  push    dx
000F  56                  push    si
0010  57                  push    di
0011  50                  push    ax
                  ;*********************************************
0012  B8 F000             mov     ax,0F000h       ;TANDY : 0E000h
                  ;*********************************************
                  ;       XY-Koordinaten RAM-Adresse ermitteln
0015  8E C0               mov     es,ax
0017  8B 76 06            mov     si,[bp+6]       ;X-Koord
001A  83 FE 00            cmp     si,0            ;X<0 ?
001D  7C 36               jl      stopp           ;dann Abbruch
001F  81 FE 027F          cmp     si,639          ;X>639 ?
0023  7F 30               jg      stopp           ;dann Abbruch
0025  8B 7E 04            mov     di,[bp+4]       ;Y-Koord
0028  83 FF 00            cmp     di,0            ;Y<0 ?
002B  7C 28               jl      stopp           ;dann Abbruch
                  ;*********************************************
002D  81 FF 015D          cmp     di,349  ;Y>349 ? TANDY: 399 !!!
                  ;*********************************************
0031  7F 22               jg      stopp           ;dann Abbruch
0033  8B DF               mov     bx,di           ;BX:=Y
0035  B1 06               mov     cl,6
0037  D3 E3               shl     bx,cl           ;BX:=Y*64
0039  8B C7               mov     ax,di
003B  B1 04               mov     cl,4
003D  D3 E0               shl     ax,cl           ;AX:=Y*16
003F  03 D8               add     bx,ax           ;BX:=Y*80
0041  8B C6               mov     ax,si           ;AX:=X
0043  B1 03               mov     cl,3
0045  D3 E8               shr     ax,cl           ;AX:=X div 8
0047  03 D8               add     bx,ax           ,BX:=Byte des
0049  B9 0007             mov     cx,7            ;Punktes
004C  23 CE               and     cx,si           ;CX:=X mod 8
004E  B0 80               mov     al,80h          ;7-(x mod 8)Bit
0050  D2 E8               shr     al,cl           ;selektieren
0052  26: 08 07           or      es:[bx],al      ;Punkt setzen
0055  58          stopp:  pop     ax  ;Registerinhalte wieder
0056  5F                 pop     di  ;herstellen
0057  5E                 pop     si
0058  5A                 pop     dx
0059  59                 pop     cx
005A  5B                 pop     bx
005B  8B E5               mov     sp,bp
005D  5D                 pop     bp
005E  C2 0004             ret     4
                  ;-----------------------------------------------
0061  55          radi:   push    bp              ;Punkt loeschen
0062  8B EC               mov     bp,sp
0064  53                  push    bx
0065  51                  push    cx
0066  52                  push    dx
0067  56                  push    si
0068  57                  push    di
0069  50                  push    ax
                  ;*********************************************
006A  B8 F000             mov     ax,0F000h       ;TANDY : 0E000h
                  ;*********************************************
006D  8E C0               mov     es,ax
006F  8B 76 06            mov     si,[bp+6]
0072  83 FE 00            cmp     si,0
0075  7C DE               jl      stopp
0077  81 FE 027F          cmp     si,639
007B  7F D8               jg      stopp
007D  8B 7E 04            mov     di,[bp+4]
0080  83 FF 00            cmp     di,0
0083  7C D0               jl      stopp
                  ;*********************************************
0085  81 FF 015D          cmp     di,349  ;Y>349 ? TANDY: 399 !!!
                  ;*********************************************
0089  7F CA               jg      stopp
008B  8B DF               mov     bx,di
008D  B1 06               mov     cl,6
008F  D3 E3               shl     bx,cl
0091  8B C7               mov     ax,di
0093  B1 04               mov     cl,4
0095  D3 E0               shl     ax,cl
0097  03 D8               add     bx,ax
0099  8B C6               mov     ax,si
009B  B1 03               mov     cl,3
009D  D3 E8               shr     ax,cl
009F  03 D8               add     bx,ax
00A1  B9 0007             mov     cx,7
00A4  23 CE               and     cx,si
00A6  B0 80               mov     al,80h
00A8  D2 E8               shr     al,cl
00AA  F6 D0               not     al
00AC  26: 20 07           and     es:[bx],al      ;Punkt loeschen
00AF  EB A4               jmp     stopp
                  ;-----------------------------------------------
00B1  55          test:   push    bp              ;Punkt testen
00B2  8B EC               mov     bp,sp
00B4  83 EC 0A            sub     sp,10
00B7  53                  push    bx
00B8  51                  push    cx
00B9  52                  push    dx
00BA  56                  push    si
00BB  57                  push    di
00BC  50                  push    ax
                  ;*********************************************
0012  B8 F000             mov     ax,0F000h       ;TANDY : 0E000h
                  ;*********************************************
00C0  8E C0               mov     es,ax
00C2  8B 76 06            mov     si,[bp+6]       ;X-Koord
00C5  83 FE 00            cmp     si,0
00C8  7C 36               jl      raus
00CA  81 FE 027F          cmp     si,639
00CE  7F 30               jg      raus
00D0  8B 7E 04            mov     di,[bp+4]
00D3  83 FF 00            cmp     di,0
00D6  7C 28               jl      raus
                  ;*********************************************
002D  81 FF 015D          cmp     di,349  ;Y>349 ? TANDY: 399 !!!
                  ;*********************************************
```

Josef Ellenrieder

Datenbank-systeme

Teil 2: Welches Datenbanksystem ist das richtige?

Bevor man sich mit den Details verschiedener Datenbanksysteme auseinandersetzt, ist eine Analyse des künftigen Einsatzgebietes nötig. Der zweite Teil dieser dreiteiligen Artikelserie beginnt deshalb mit Fragen, die sich der künftige Datenbankanwender stellen sollte.

Welche betrieblichen Bereiche sind betroffen? Welche Daten müssen erfaßt werden, und welche Informationen soll das System liefern? Wie groß wird die zu verarbeitende Datenmenge sein? Es ist durchaus von Bedeutung, ob 500 oder 5000 Adressen verwaltet werden müssen.
Auch die zukünftige Entwicklung muß in diese Analyse mit einbezogen werden. Was hilft es, wenn nach einem Jahr das Datenbanksystem aus den Nähten platzt bzw. die Obergrenze der verwaltbaren Datenmenge erreicht wird oder die Zugriffe minutenlang dauern.

Fragen über Fragen

Ebenso muß geklärt werden, welche Beziehungen zwischen den Daten bestehen sollen. Welche Daten müssen wann, wie und wo verarbeitet werden? Welche Daten dienen Verarbeitungs- und welche Informationszwecken? Wie schnell muß auf die Daten zugegriffen werden können – eventuell, um sie weiterzuverarbeiten?
Sind alle diese Fragen beantwortet, sollte man sich auch Gedanken darüber machen, wie das neue Datenbanksystem in die bestehende Organisation eingegliedert werden soll und welche Mitarbeiter sich damit beschäftigen werden.
Erst nach dieser Analyse sollte man darangehen, die Vorzüge der verschiedenen Datenbanksysteme miteinander zu vergleichen. Jetzt hat man Informationen, um sagen zu können, ob ein Datenbanksystem die anfallenden Datenmengen überhaupt verwalten kann.
In diesem Zusammenhang ist es auch sehr interessant, die Angaben bezüglich der Verwaltungsgeschwindigkeit zu vergleichen. Wie lange dauert bei einer größeren Datenmenge der Zugriff auf bestimmte Daten, wie lange ein Sortiervorgang oder eine bestimmte Auswertung? Die Antworten auf diese Fragen sind natürlich auch durch die benutzte Hardware bestimmt. Die Zugriffsgeschwindigkeit bei Disketten ist aufgrund der unterschiedlichen Bauweise nun mal geringer als bei Festplatten.

Geringe Daten-Redundanz

Eine der Hauptanforderungen an ein Datenbanksystem ist die möglichst geringe Redundanz bei der Datenspeicherung, d. h. alle Daten sollen möglichst nur einmal physisch auf dem Speichermedium vorhanden sein. Die Daten-Redundanz hängt in erster Linie von der systeminternen Speicherungs- und Zugriffsform ab. Dieser Punkt kann aber nur von einem Fachmann beurteilt werden, der über die entsprechenden systeminternen Informationen verfügt.
Die meisten Systeme lösen dieses Problem geschickt. Wiederholte Speicherung von Daten tritt nur dann auf, wenn sich der Anwender bei der Definition der Daten ungeschickt verhält. Es ist zum Beispiel günstiger, Herstelleranschrift und -namen nur einmal in einer Relation (Satz) zu erfassen und die Verbindung zu den Herstellerdaten in anderen Relationen (Sätzen) durch Einführen einer Herstellernummer zu schaffen.

Anpaßbarkeit und Datenunabhängigkeit

Ein wichtiger Anspruch ist Flexibilität. Informationszusammenhänge sind immer Veränderungen unterworfen. Somit muß auch die Datenbank, die die Informationen liefern soll, ein flexibles Gebilde sein. Die Datenbank muß an neue Anforderungen angepaßt werden können.
Die häufigsten Änderungen sind Ergänzungen neuer Felder in bestehenden Informationseinheiten. Erweitern von Feldern, Einrichten neuer Beziehungen zwischen den Daten und Schaffen neuer Auswertungsmöglichkeiten, z. B. Tabellen, Statistiken oder Grafiken. Diese Anpassungen sollen mit vertretbarem Aufwand und der Garantie, daß keine bereits erfaßten Daten verlorengehen, durchführbar sein.

Die Anpaßbarkeit ist vor allem gegeben, wenn es sich um ein relationales Datenbanksystem handelt. Da bei relationalen Datenbanksystemen keine physische Verbindung und keine Strukturierung der Daten vorhanden sind, kann jederzeit eine neue Beziehung eingeführt werden. Es ist eine typische Eigenschaft relationaler Systeme, nach dem Inhalt sämtlicher Felder suchen zu können. Somit macht es auch keine Probleme, neue Auswertungen zu treffen.
Unbedingt erforderlich ist eine hohe Datenunabhängigkeit. Die eben genannten Anpassungen sollten möglichst wenig Änderungen in bestehenden Anwendungsprogrammen nach sich ziehen. Der

Entscheidungskriterien

- Einsatzbereich des Datenbanksystems im Unternehmen
- Art der Daten
- Datenmenge
- Beziehungen zwischen den Daten
- Künftige Entwicklung

Anforderungen an ein Datenbanksystem

- Verarbeitung großer Datenmengen, günstige Antwortzeit
- Geringe Datenredundanz
- Flexibilität: Anpaßbarkeit, Datenunabhängigkeit
- Reorganisationsmöglichkeiten
- Schutz vor Zerstörung der Daten
- Maßnahmen zur Datensicherung und Wiederherstellung
- Schutz gegen unberechtigten Zugriff
- Synchronisation paralleler Zugriffe
- Schnittstellen
- Interaktive Abfragesprache
- Reportgenerator
- Eigendokumentation
- Übersichtliche Handbücher
- Eingebaute Hilfsfunktion

beit, ist aber unflexibel gegenüber Wünschen, die von Standardarbeiten abweichen.
Für komplexere Anwendungen ist es häufig günstiger, die Kommunikation mit der Datenbank über Programme abzuwickeln. Zu diesem Zweck kann man entweder die Befehle der Abfragesprache zu Programmen zusammenfassen, oder das System hat eine Schnittstelle zu herkömmlichen Programmiersprachen. Die erste Methode ist die weiter verbreitete. Das setzt aber voraus, daß es in dieser Abfragesprache neben den interaktiven Befehlen auch Befehle zur Gestaltung von Programmverzweigungen, Programmschleifen und ähnliches gibt.

Reportgeneratoren

Es kann äußerst nützlich sein, wenn es auf einfache Art und Weise möglich ist, die Daten ordentlich formatiert mit Überschriften in einer Liste auf den Bildschirm oder Drucker zu bringen. Die meisten Datenbanksysteme stellen dazu einen sogenannten Reportgenerator zur Verfügung. Die Flexibilität und Handhabungsweise dieses Reportgenerators hat natürlich großen Einfluß auf die Effizienz der gesamten Arbeit (*Bilder 11* bis 13).
Vor allem im Zusammenhang mit der Erstellung von Reports ist es nützlich, wenn das System sozusagen eine Eigendokumentation kennt. Mit Hilfe von Zusatzprogrammen wird angezeigt, welche Felder beispielsweise mit welchen Eigenschaften in einer Datenbank abgelegt worden sind.

Dokumentation

Große Unterschiede weisen die verfügbaren Datenbanksysteme hinsichtlich der Qualität und Art ihrer Begleitliteratur auf. Ein unübersichtliches und unvollständiges Handbuch kann jeden Benutzer zur Verzweiflung bringen. Vor allem der mit Datenbanken Unerfahrene sollte darauf achten, wie verständlich die Handbücher sind.

Besonders hilfreich für die Einarbeitung und auch weitere Beschäftigung mit dem System ist eine eingebaute Hilfefunktion, über die während der Arbeit am Bildschirm Systeminformationen, z. B. eine Liste aller Befehle, abgerufen werden können. Sie erspart Blättern und Suchen in den Handbüchern. Sicherlich ist es für Datenbanken auf dem PC-Markt schwer, die hier aufgeführten Kriterien zu erfüllen. Diese Anforderungen gehen von Datenbanksystemen für Großcomputer aus, die in jahrelanger Entwicklung entstanden sind und in einer völlig anderen Preisklasse liegen.
Trotzdem bleiben die Anwendungen und Probleme, ob Großrechner oder Kleincomputer, im Normalfall dieselben. Und wenn sich eine Software schon Datenbanksystem nennt, muß es auch dessen Eigenschaften aufweisen.

Routine zum Übernehmen von Dateinamen in Strings

Das Programm Filestring (*Bild*) übernimmt aus dem Befehl FILES Dateinamen, um diese in Strings zur Belegung von Funktionstasten umzusetzen. In dem Beispielprogramm wird in einem Subdirectory nach Dateinamen mit der Ergänzung KAT gesucht. Diese werden dann auf die Funktionstasten gelegt. In Zeile 40 wird der Bildschirm gelöscht, der Cursor auf Zeile 11, Spalte 1 positioniert, um eine definierte Ausgangsposition zu erreichen. Der FILES-Befehl gibt dann das Inhaltsverzeichnis auf dem Bildschirm aus. In Zeile 50 wird eine Variable für das Lesen vom Bildschirm initialisiert (Zeile aus LOCATE-Befehl + 1). Die Zeilen 60–120 enthalten die FOR-NEXT-Schleife für das Lesen des Inhaltsverzeichnisses vom Bildschirm. In Zeile 70 wird der Variablen Y der ASCII-Wert der Position BZ,ZZ zugewiesen, worauf in Zeile 80 führende Leerzeichen abgeschnitten werden. Die Befehle in Zeile 90 bauen den Textstring auf (TE$(N)), in Zeile 100 werden die Dateien . <DIR> und .. <DIR> abgeschnitten. Sind die Dateinamen vollständig, wird in Zeile 110 die Variable N weitergeschaltet. In dieser Zeile können die Endungen, die gefragt sind, eingesetzt werden. In Zeile 130 wird die Bildschirmzeile verändert (erhöht).
In den Zeilen 190–210 werden schließlich die Funktionstasten mit den Textstrings belegt (da die Felder mit TE$() von Null an laufen und die KEY-Tasten von 1 an, wird in 200 jeweils eine Eins aufaddiert, d. h. der Text TE$(0) wird auf die Funktionstaste F1 gelegt).
Dieses Programm wird als Unterprogramm in einem Eingabeprogramm verwendet, welches Zugriff auf verschiedene sich öfter ändernde Dateien hat. Dadurch kann es nicht so gemacht werden, daß hier über eine Installationsdatei der Name eingegeben werden kann, da auch die Anzahl der Dateien unterschiedlich ist.

Wolfgang Prestel

```
10 REM aus FILES in GWBASIC Strings zum Belegen der F-Tasten machen
20 REM es wird nach Dateien mit Erweiterung .KAT in einer SUB-Directory gesucht
30 REM Copyright (C) Wolfgang Prestel, Sommerhofenstr.34, 7032 Sindelfingen
40 CLS:LOCATE 11,1:FILES
50 BZ=12
60 FOR ZZ=1 TO 80
70 Y=SCREEN (BZ,ZZ)
80 IF LEN(TE$(N))=0 AND Y=32 THEN 120
90 TE$(N)=TE$(N)+CHR$(Y)
100 IF RIGHT$(TE$(N),5)="<DIR>" THEN TE$(N)="":GOTO 120
110 IF RIGHT$(TE$(N),4)=".KAT" THEN N=N+1
120 NEXT ZZ
130 BZ=BZ+1
140 IF N=NA THEN 150 ELSE NA=N:GOTO 60
150 IF INSTR(TE$(N),"Bytes free")>0 THEN TE$(N)="":N=N-1
160 FOR ZZ= 0 TO N
170 TE$(ZZ)=RIGHT$(TE$(ZZ),12)
180 NEXT ZZ
190 FOR S=0 TO 9
200 KEY 1+S,TE$(S)
210 NEXT S
```

Dieses GW-Basic-Programm übernimmt Dateinamen in Strings

Martin Brendle

Wahrheitstafeln für Logik-PROMs

Ein Turbo-Pascal-Programm erleichtert den Schaltungs-Entwurf

Mit programmierbaren Nur-Lese-Speichern (PROMs) lassen sich Digital-Schaltungen in vielen Fällen einfach aufbauen. Die gewünschten logischen Verknüpfungen lassen sich mit einem Turbo-Pascal-Programm in die Wahrheitstafel des PROM-Bausteins umsetzen.

Gegenüber einem PAL-Baustein (PAL = Programmable Array Logic) bietet ein PROM den Vorteil, daß beliebige Funktionen realisiert werden können. Die Anzahl der oder-verknüpften Produkt-Terme ist nicht beschränkt. Allerdings steigt die erforderliche PROM-Kapazität exponentiell mit der Zahl der Eingangsvariablen: Die Anzahl der Adresseneingänge des PROM-Bausteins gibt die Maximalzahl der zulässigen Eingangsvariablen an, die Wortlänge die Maximalzahl der Ausgangsvariablen.

Eine kombinatorische Logik-Schaltung kann man durch eine Tabelle beschreiben, die jeder möglichen Kombination von Zuständen der Eingangsvariablen (jedem Eingangszustand) die Werte der Ausgangsvariablen (den Ausgangszustand) zuordnet, die „Wahrheitstafel". Bei einer Implementation durch einen Nur-Lese-Speicher (PROM) entspricht jedem Eingangszustand eine Adresse; der zugehörige Ausgangszustand wird in der zu dieser Adresse gehörenden Speicherzelle gespeichert. Die Wahrheitstafel entspricht also den im PROM-Baustein abzulegenden Daten.

Im allgemeinen liegen die Anforderungen an eine Logikschaltung primär nicht als Wahrheitstafel, sondern als eine Menge logischer Beziehungen zwischen den Eingangs- und Ausgangsvariablen vor. Abgesehen von ganz simplen Fällen ist eine manuelle Erzeugung der PROM-Daten wegen des hohen Zeitaufwandes und wegen der Fehleranfälligkeit nicht zu empfehlen. An Hand eines Beispiels zeigen wir, wie diese Aufgabe mit einem PASCAL-Programm gelöst werden kann.

Bild 1 zeigt die Schaltung zur Erzeugung des Signale /RAS0, /RAS1, /RAS2 und /RAS3 in einem PC. Jedes dieser Signale selektiert eine der vier Speicher-Bänke. Ein 256 × 4-Bit-PROM erzeugt aus den Adreßsignalen XA16, XA17, XA18 und XA19 die Auswahl-Signale BANKA und BANKB, welche die binär codierte Banknummer darstellen, und das Freigabe-Signal RAM, welches bei jeder gültigen Hauptspeicher-Adresse aktiv wird. Ein weiteres Eingangs-Signal JP3 wird mit einer Drahtbrücke eingestellt. Damit kann eine von zwei Speicher-Bestükkungen gewählt werden:

- vier Bänke mit 64-KBit-DRAMs
- zwei Bänke (die beiden ersten) mit 256-KBit-DRAMS und zwei Bänke mit 64-KBit-DRAMs.

Mit einem neuen PROM ist es möglich, die drei ersten Bänke mit 256-KBit-DRAMs zu bestücken. Die verfügbare Gesamtkapazität kann mit der vorhande-

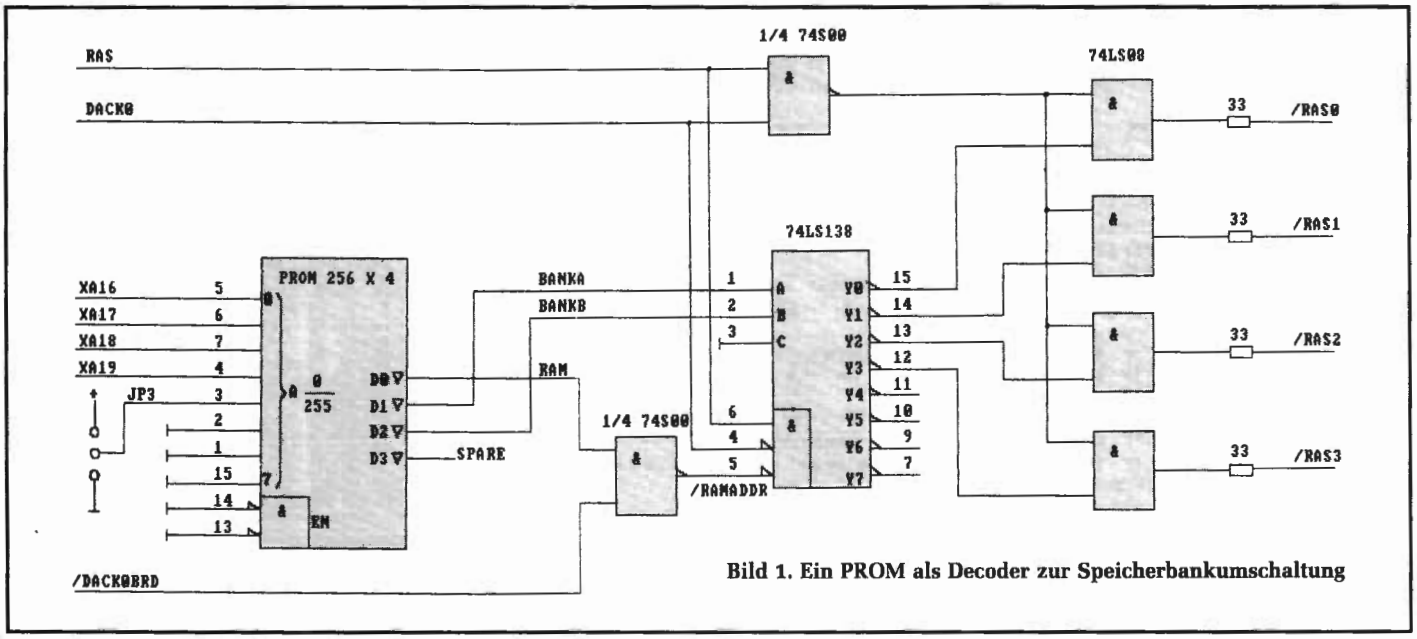


Bild 1. Ein PROM als Decoder zur Speicherbankumschaltung

```
(*A*)  PROGRAM WTAFEL(PROM);

(*     Programm zur Erzeugung der Wahrheitstafel für ein Logik-PROM      *)
(*     Die Ausgangsdaten werden im "ASCII HEX SPACE"-Format in einer     *)
(*     Datei mit dem durch die Konstante FILENAME definierten Namen      *)
(*     gespeichert.                                                      *)

       CONST
(*A  Name der Ausgangsdatei:           *)     FILENAME = 'WTAFEL.HEX';
(*A  Anzahl der PROM-Adresseneingänge: *)     ADRINPNR = 8;
(*A  Anzahl der PROM-Datenausgänge:    *)     DATOUTNR = 4;

       TYPE
       INT4    = ARRAY[1..4] OF INTEGER;
       ITYP    = ARRAY[1..ADRINPNR] OF BOOLEAN;
       OTYP    = ARRAY[1..DATOUTNR] OF BOOLEAN;
       STR16   = PACKED ARRAY[1..16] OF CHAR;

       VAR
       ADDRESS, CELLNR, DIGNR,
       AUXI1, AUXI2, AUXI3:          INTEGER;
       I:                            ITYP;
       O:                            OTYP;
       PROM:                         TEXT;
       HEXDIG:                       STR16;
       HEXNR:                        INT4;

(*     Namen der Eingangs- und Ausgangs-Signale:                         *)
(*A)   XA16, XA17, XA18, XA19,
(*A)   JP3, A5, A6, A7,
(*A)   RAM, BANKA, BANKB, SPARE:     BOOLEAN;

(*A    Hilfs-Variable:                                                   *)
(*A)   VALID:                        BOOLEAN;

       BEGIN
(*     Der Vorspann wird erzeugt:                                        *)
       ASSIGN(PROM,FILENAME); REWRITE(PROM); WRITE(PROM,CHR(2),'$A0000,');

(*     Hier wird eine Tabelle mit den Hexadezimalziffern aufgesetzt.     *)
       HEXDIG := '0123456789ABCDEF';

(*     Hier wird die Anzahl der PROM-Speicherzellen berechnet.           *)
       CELLNR := 1;
       FOR AUXI1 := 1 TO ADRINPNR DO
         CELLNR := CELLNR + CELLNR;

(*     Wieviele Hexadezimal-Ziffern braucht man für eine Speicherzelle?  *)
(*     (1, 2, 3 oder 4)                                                  *)
       DIGNR := (DATOUTNR + 3) DIV 4;

(*     Berechnung der PROM-Inhalte:                                      *)
       FOR ADDRESS := 0 TO (CELLNR - 1) DO
         BEGIN
         AUXI1 := 16;
         IF DIGNR > 2 THEN AUXI1 := 8;
         IF (ADDRESS MOD AUXI1) = 0
           THEN WRITELN(PROM)
           ELSE
             BEGIN
             IF (ADDRESS MOD 8) = 0 THEN WRITE(PROM,'  ');
             END;
         AUXI1 := ADDRESS;
         FOR AUXI2 := 1 TO ADRINPNR DO
           BEGIN
           AUXI3 := AUXI1 MOD 2;
           AUXI1 := AUXI1 DIV 2;
           IF AUXI3 = 1 THEN I[AUXI2] := TRUE
                        ELSE I[AUXI2] := FALSE;
           END;

(*     Definitionen, welche die Verdrahtung der PROM-Eingänge und die    *)
(*     aktiven Pegel der Eingangssignale betreffen:                      *)
(*A)    XA16 := I[1];
(*A)    XA17 := I[2];
(*A)    XA18 := I[3];
(*A)    XA19 := I[4];
(*A)    JP3  := I[5];
(*A)    A5   := I[6];
(*A)    A6   := I[7];
(*A)    A7   := I[8];

(*     Logische Beziehungen zwischen Eingangs- und Ausgangs-Signalen:    *)
(*A)    VALID  := NOT A5 AND NOT A6 AND NOT A7;
(*A)    RAM    := VALID AND (NOT XA19 OR (NOT XA18 AND
(*A)              (NOT XA17 OR (JP3 AND NOT XA16))));
(*A)    BANKA  := XA18 OR NOT RAM;
(*A)    BANKB  := XA19 OR NOT RAM;
(*A)    SPARE  := FALSE;

(*     Definitionen, welche die Verdrahtung der PROM-Ausgänge und die    *)
(*     aktiven Pegel der Ausgangssignale betreffen:                      *)
(*A)    O[1] := RAM;
(*A)    O[2] := BANKA;
(*A)    O[3] := BANKB;
(*A)    O[4] := SPARE;

(*     Berechnung eines PROM-Wortes aus den Ausgangsvariablen:           *)
       AUXI1 := 0;
       FOR AUXI2 := DATOUTNR DOWNTO 1 DO
         BEGIN
         AUXI1 := AUXI1 + AUXI1;
         IF O[AUXI2] = TRUE THEN AUXI1 := AUXI1 + 1;
         END;
       FOR AUXI3 := 1 TO DIGNR DO
         BEGIN
         HEXNR[AUXI3] := AUXI1;
         AUXI1 := AUXI1 DIV 16;
         HEXNR[AUXI3] := HEXNR[AUXI3] - (16 * AUXI1);
         END;

(*     Ausgabe:                                                          *)
       FOR AUXI1 := DIGNR DOWNTO 1 DO
         BEGIN
         AUXI3 := 1 + HEXNR[AUXI1];
         WRITE(PROM,HEXDIG[AUXI3]);
         END;
       WRITE(PROM,' ');
       END;

(*     Nachspann:                                                        *)
       WRITELN(PROM);  WRITE(PROM,CHR(3));  CLOSE(PROM);
       END.
```

Bild 2.
Das Turbo-Pascal-Programm

nen Drahtbrücke auf 640 KByte oder 704 KByte eingestellt werden. Ist der Adreßbereich von A0000H bis AFFFFH anderweitig belegt – etwa durch eine EGA-Karte – so darf der Speicherbereich über 640 KByte nicht verwendet werden.

Das Turbo-Pascal-Programm (*Bild 2*) kann für andere Anwendungen abgeändert werden. Die in Frage kommenden Stellen sind durch mit A beginnende Kommentare gekennzeichnet. Die Konstante FILENAME bestimmt den Namen der Ausgangsdatei. Die PROM-Konfiguration wird durch die Konstanten ADRINPNR und DATOUTNR festgelegt. ADRINPNR gibt die Zahl der Adreßeingänge und damit auch die Zahl der Eingangsvariablen (=Eingangssignale) an. DATOUTNR gibt die Zahl der Datenausgänge und damit die Zahl der Ausgangsvariablen (=Ausgangssignale) an. Ein PROM, dessen Speicher in 256 × 4 Bit organisiert ist, hat acht Eingänge (Adreßleitungen) und vier Ausgänge. In unserem Beispiel sind dementsprechend die Variablen ADRINPNR und DATOUTNR auf diese Werte gesetzt.

Ein Definitions-Block legt die Namen von Eingangs- und Ausgangs-Signalen fest, ein weiterer die Zuordnung der Eingangssignale zu den Eingangs-Anschlüssen und die aktiven Pegel der Eingangssignale, ein dritter nimmt die entsprechenden Zuordnungen für die Ausgangssignale vor. Darüberhinaus müssen in einer Gruppe von Anweisungen die logischen Beziehungen zwischen den Eingangs- und Ausgangs-Signalen definiert werden. Manchmal kann man sich hierbei durch die Benutzung von Hilfsvariablen Schreibarbeit sparen. Hilfsvariable, in unserem Beispiel VALID, müssen im Variablen-Deklarationsteil angegeben werden. *Bild 3* zeigt die vom Programm WTAFEL erzeugte Datei. Diese Datei im ASCII-Hex-Space-Format stellt den Hex-Dump des Speicherinhalts des PROMs von Bild 1 dar. Das ASCII-Hex-Space-Format wird von Kontron-Programmiergeräten verstanden. Das erste Zeichen in diesem Format ist das ASCII-Steuerzeichen STX (Start of Text = 02H), das auf unserem Ausdruck als B dargestellt wird. $A0000 bedeutet, daß das Programmiergerät bei Adresse $0000 des PROMs mit der Programmierung beginnen soll. Das Leerzeichen dient als Begrenzer zwischen den einzelnen Daten; das Programmgerät inkrementiert bei jedem Leerzeichen seinen internen Adreßzähler um eins.

Am Ende der Datei steht das ASCII-Zeichen ETX (End of Text = 03H), das auf unserem Ausdruck als C dargestellt wird.

```
B$A0000,
1 1 1 1 3 3 3 3   5 5 6 6 6 6 6 6
1 1 1 1 3 3 3 3   5 5 5 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6   6 6 6 6 6 6 6 6
C
```

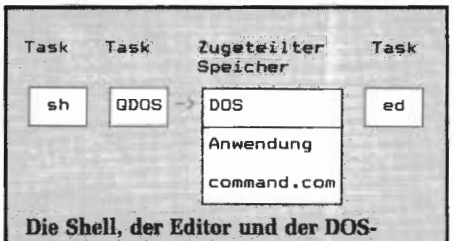
Bild 3. Der Hex-Dump des PROMs von Bild 1

Echtzeit-Betriebssystem für den PC

Ein Unix-ähnliches Betriebssystem – speziell für den PC und den AT – wurde vom kanadischen Hersteller Quantum entwickelt. QNX, so lautet der Name dieses Betriebssystems, ist für Echtzeitanwendungen geeignet. Die Eigenschaften des 80286-Prozessors, insbesondere der „protected mode", werden voll ausgenutzt, wenn QNX auf Rechnern der AT-Klasse installiert wird. Bis zu 40 Tasks können quasi simultan ablaufen. Nach dem Booten laufen vier System-Tasks:
- task administrator
- file system administrator
- device administrator
- idle administrator

Bei der netzwerkfähigen Version kommt noch der network administrator hinzu. Somit bleiben für den Anwender noch 35 Tasks frei.
QNX kennt zur Taskverwaltung 15 Prioritätsebenen. Vom task administrator werden Tasks erzeugt, gelöscht und die Speicherplatzverwaltung durchgeführt. Der task administrator hat die höchste Priorität (Priorität 1).
Verantwortlich für die Verwaltung des Datei-Systems ist der file system administrator (Priorität 3).

Über den device administrator laufen alle Zugriffe auf zeichenorientierte Betriebsmittel (Drucker, Terminal). Diese Task hat die zweithöchste Priorität im System.



Die Shell, der Editor und der DOS-Emulator laufen simultan

Die Task des idle administrator läuft nur, wenn keine andere Task aktiv ist. In der netzwerkfähigen Version von QNX führt der network administrator die gesamte Kommunikation über das Netzwerk durch. Eine Task kann mit jeder anderen Task auf jeder beliebigen Maschine innerhalb des Netzwerks kommunizieren.
Entscheidend für die Leistungsfähigkeit eines Echtzeit-Betriebsystems ist das schnelle Umschalten zwischen den einzelnen Tasks. Auf einem 8 MHz AT können bis zu 2800 Task-Umschaltungen pro Sekunde stattfinden.

QNX belegt ungefähr 100 KByte Speicher für eine Shell. In einem Einbenutzer-System wird mindestens 192 KByte RAM benötigt. Sollen Kommandos im Hintergrund ablaufen, so muß der PC mit mindestens 256 KByte RAM ausgestattet sein. Auf dem AT kann QNX bis zu 15 MByte RAM verwalten.
Mit dem DOS-Emulator QDOS von Quantum kann eine DOS-Anwendung unter QNX als Task laufen, vorausgesetzt die DOS-Anwendung ist „sauber" programmiert (Bild). Das System stürzt jedoch ab, wenn das DOS-Anwenderprogramm

- außerhalb des ihm zugewiesenen Speicherbereichs zugreift oder
- Software- oder Hardware-Interrupts ändert.

Intern verwendet QNX die Software-Interrupts 68H...7FH und F0H...FFH.

Für QNX gibt es bereits einen C- , einen Basic-Compiler und verschiedene Anwenderprogramme.
Besonders empfehlenswert ist QNX in Netzwerken und technischen Anwendungen, für die ein Echtzeit-Betriebssystem benötigt wird.

Nach Unterlagen von repas, Dreieich

Holger Schmidt

Schneller Drawline-Algorithmus

Noch immer gibt es Computer, die standardmäßig nicht über die wichtigsten Grafikbefehle verfügen oder bei denen man feststellt, daß zwar Grafiklösungen vorhanden sind, deren Geschwindigkeitsverhalten aber nicht optimal ist. Beispielhaft sei hier Turbo-Pascal erwähnt, für das, sofern es nicht auf einem IBM-PC oder Kompatiblen eingesetzt wird, keine Grafikbefehle zur Verfügung stehen. Es bleibt einem nichts anderes übrig, als sich entweder ein entsprechendes Grafik-Paket zu besorgen oder aber selbst zur Tastatur zu greifen.

Am Anfang einer Grafikbibliothek stehen meist Befehle wie Einschalten des Grafikbildschirms, Löschen der Grafikseite, Setzen eines Punktes und Zeichnen einer Linie. Das Löschen ist zumindest bei einem memory-mapped Grafikspeicher unproblematisch (Man lösche einfach den entsprechenden Speicherbereich). Das Aktivieren des Grafikbildschirmes und das Setzen eines Punktes hingegen sind rechnerspezifisch; eine allgemeine Lösung läßt sich nicht angeben. Ist jedoch ein Weg zum Setzen eines Punktes gefunden, so läßt sich allgemein ein „Drawline-Algorithmus" angeben, da man das Zeichnen einer Linie als Setzen von Punkten nach einem bestimmten Verfahren verstehen kann. Doch wie so oft führen auch hier viele Wege zum Ziel, gefunden werden sollte jedoch der beste oder aber zumindest ein guter Weg.

Vorgestellt werden soll daher ein aus dem sogenannten „Bresenham-Algorithmus" abgeleiteter Algorithmus zum Zeichnen einer Linie (zusammengesetzt aus einzelnen Punkten), wobei wir jedoch zum Verständnis auch die Herleitung dieses Algorithmus und eine Geschwindigkeitsoptimierung nicht vernachlässigen wollen. Einige werden jetzt sagen, „Das interessiert doch gar nicht, Hauptsache es läuft", viele andere, und ich zähle mich selbst hinzu, finden diese Einstellung unbefriedigend. Bei Benutzung eines solchen Algorithmus möchte man doch auch etwas über den Hintergrund erfahren und deshalb gehen wir hier etwas ausführlicher darauf ein.

Die Herleitung des Algorithmus gliedert sich in vier Punkte:

1. Mathematische Formulierung einer Ursprungsgeraden
2. Übertragung auf Programmierebene
3. Geschwindigkeitsoptimierung
4. Verallgemeinerung für beliebige Geraden

Aufgabe des endgültigen Algorithmus wird es sein, zwei beliebige durch ihre Koordinaten gegebene Punkte P1 (x1, y1) und P2 (x2, y2) durch eine gerade Linie verbinden.

Mathematische Formulierung einer Ursprungsgeraden

Zur Bestimmung der Funktion einer Ursprungsgeraden wie in *Bild 1* ist lediglich deren Steigung m zu ermitteln. Da uns die Koordinaten von P2 (x2, y2) bekannt sind, gilt:

$m = y2/x2$ und damit die Geradengleichung [1]

$f(x) = (y2/x2) * x$

Übertragung auf Programmierebene

Bevor wir zur eigentlichen Entwicklung schreiten, treffen wir noch eine Einschränkung, und zwar:

$y2 <= x2$, d. h. $m <= 1$

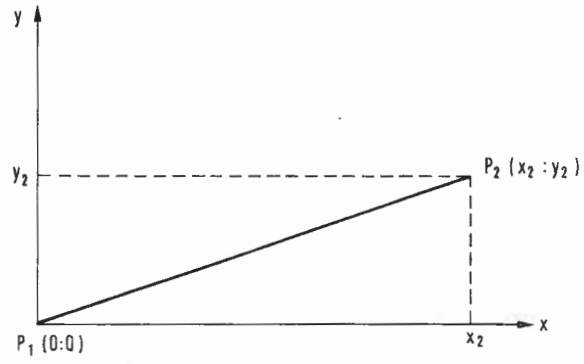
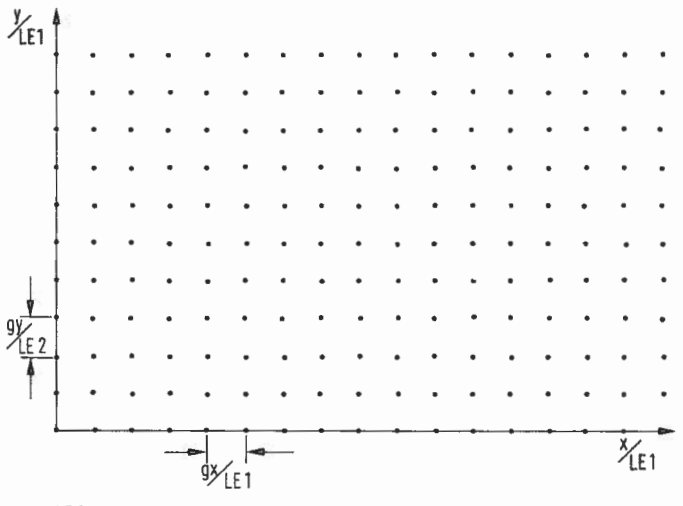


Bild 1. Die Funktion einer Geraden wird durch ihre Steigung bestimmt

Bild 2. Die Punktmatrix eines Bildschirms



LE1 : = Längeneinheit 1
LE2 : = Längeneinheit 2
gx,gy : = Gitterkonstanten (Abstand zwischen zwei Punkten)

Warum diese Einschränkung? Hierzu betrachten wir die zu zeichnende Linie genauer: Der Bildschirm ist in eine Punktmatrix, wie in *Bild 2* gezeigt, eingeteilt. Wollen wir eine Linie zeichnen, so suchen wir die Punkte, die möglichst nahe an der Geraden liegen und setzen diese. Die Genauigkeit wird also maßgeblich durch den Abstand zwischen zwei möglichen Punkten, den Gitterabstand, beeinflußt. Betrachten wir diese Punktmatrix als Koordinatensystem mit den Einheiten gx (Gitterabstand in x-Richtung) und gy (Gitterabstand in y-Richtung): Wir zeichnen eine Linie, indem wir zu jedem Punkt x (x1 < = x < = x2, x ist ein Vielfaches von gx) den Punkt y suchen, der am nächsten an der Linie liegt. Die Darstellung in *Bild 3* verdeutlicht dieses Verfahren (ein Kreuz bedeutet: Punkt gesetzt). Haben wir nun obige Einschränkung gemacht, so kann dies in einer Schleife durch Inkrementieren (≙ + 1, da gx/LE1 = 1) von x erreicht werden, bis x schließlich x2 erreicht. Was würde passieren, wenn wir diese Einschränkung nicht gemacht hätten? Verdeutlicht wird dies in *Bild 4*. Die Anzahl der Punkte zur Approximation einer Linie wäre unzureichend (Während in Bild 3 zur Approximation einer Linie 16 Punkte gesetzt werden, sind dies in Bild 4 für eine in etwa gleich lange Linie nur 7 Punkte; als Extremfall stelle man sich eine fast senkrechte Linie vor).
Unter den angegebenen Einschränkungen läßt sich eine Linie demnach wie folgt approximieren:
a) Man berechne für alle x (x1 < = x < = x2, xεQ) f(x), also den dazugehörigen Funktionswert auf der Geraden (eigentlich handelt es sich natürlich nur um eine Strecke)
b) Man berechne durch Rundung den dem berechneten Wert am nächsten gelegenen Punkt (Rundung auf ganze Zahlen, da x, yεQ).

Also gilt [2]:
Y: = ROUND (x * y2/x2),
x1 < = x < = x2, xεQ.

Prinzipiell liefert dieser Algorithmus natürlich (unter den angesprochenen Restriktionen) ein richtiges Ergebnis. Die Ausführungsgeschwindigkeit wird jedoch durch REAL-Multiplikation und REAL-Division negativ beeinflußt.

Geschwindigkeitsoptimierung

Will man den so gewonnenen Algorithmus optimieren, muß man zweifellos an dem Rundungsbefehl ansetzen. Das Ziel

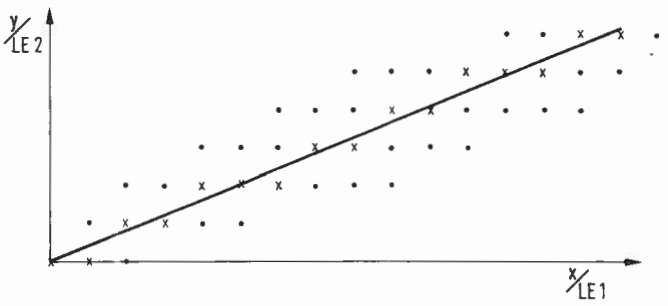


Bild 3. Eine Linie wird gezeichnet

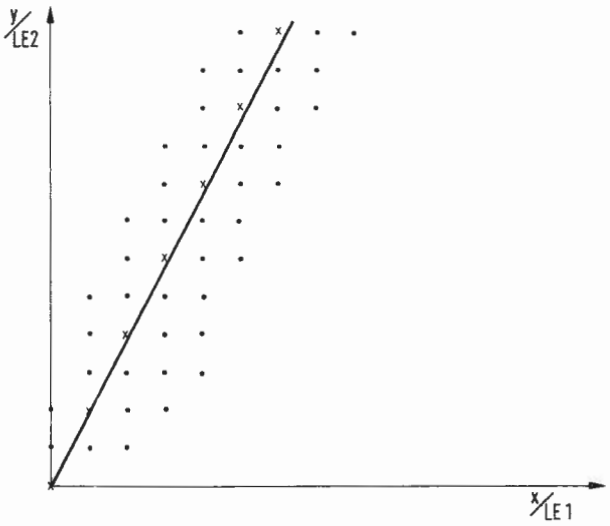


Bild 4. So würde eine Linie ohne die im Artikel beschriebenen Einschränkungen gezeichnet

```
(IF x*y2/x2-1/2 ≤ y THEN y: = y
ENDIF (Abrundung)
(IF x*y2/x2-1/2 ≥ y THEN y: = y + 1
ENDIF (Aufrundung)

(IF x*y2-1/2*x2 ≤ y*x2 THEN y: = y
ENDIF
(IF x*y2-1/2*x2 ≥ y*x2 THEN
y: = y+1 ENDIF

(IF 2*x*y2-x2-2*y*x2 ≤ 0 THEN
y: = y ENDIF
(IF 2*x*y2-x2-2*y*x2 ≥ 0 THEN
y: = y+1 ENDIF

(e: = 2*x*y2-x2-2*x2*y
(IF e ≤ 0 THEN y: = y ENDIF
(IF e ≥ 0 THEN y: = y+1 ENDIF
(e: = 2*y2*(x+1)-x2-2*x2*y
(Inkrementierung von x)
(Die erste Zeile stellt eine Anfangsbedingung dar. Die folgenden drei Zeilen werden innerhalb einer Schleife durchlaufen)

(e: = 2*x*y2-x2-2*x2*y
(IF e ≤ 0 THEN e: = 2*y2*(x+1)
-2*x2*y-x2 : y: = y ENDIF

(IF e ≥ 0 THEN e: = 2*y2*(x+1)
-2*x2*(y+1)-x2 : y: = y+1 ENDIF

(e: = 2*x*y2-2*x2*y-x2
(IF e ≤ 0 THEN e: = e+2*y2 ENDIF
(IF e ≥ 0 THEN y: = y+1 ;
e: = e+2*y2-2*x2 ENDIF

Somit ergibt sich grob skizziert folgender Algorithmus:
a: = 2*y2
b: = 2*x2-a

e: = a-x2
(Erster Schritt: x = 1; y = 0)
(x,y): = (0,0) (Anfangspunkt P1 (0,0))
PLOT (x,y)
WHILE x≤x2 LOOP
x: = x+1
IF e ≤ 0 THEN e: = e+a ENDIF
IF e ≥ 0 THEN y: = y+1 ; e: = e-b ENDIF
PLOT (x,y)
ENDLOOP

Die beiden IF-Anweisungen können noch wie folgt ersetzt werden:

IF e ≥ 0 THEN y: = y+1 ; e: = e-b
ELSE e: = e+a
ENDIF
```

Bild 5. Der Algorithmus wird optimiert

```
Procedure Line (x1, y1, x2, y2, farbe : INTEGER);
{ Die Prozedur verbindet die Punkte P1(x1,y1) und P2(x2,y2) mit der in    }
{ 'farbe' spezifizierten Zeichenfarbe.                                    }

VAR x1, y1, x, y,                                  { Hilfsvariablen       }
    dx, dy,
     a,  b, e,
    help,                                          { Variablentausch      }
    ende                 : INTEGER;                { Kennzeichnung Linienende }
    winkelhalbierende    : BOOLEAN;

BEGIN { Prozedur 'Line' Anfang }

   { Defaultwerte für Variablen }
   x1 := 1; y1 := 1;
   x  := x1; y := y1;
   ende := x2;
   dx   := x2 - x1;
   dy   := y2 - y1;
   winkelhalbierende := False;

   { Spiegelung an y-Achse }
   IF dx < 0 THEN
      BEGIN
         dx := -dx;
         x1 := - 1;
      END;

   { Spiegelung an x-Achse }
   IF dy < 0 THEN
      BEGIN
         dy := -dy;
         y1 := - 1;
      END;

   { Spiegelung an Winkelhalbierender des 1. Quadranten }
   IF dx < dy THEN
      BEGIN
         help := x;  x := y;  y := help;
         help := dx; dx := dy; dy := help;
         help := x1; x1 := y1; y1 := help;
         ende := y2;
         winkelhalbierende := True;
      END;

   { Hilfsvariablen }
   a := dy shl 1;
   b := dx shl 1 - a;
   e := a - dx;

   { Zeichnen der Geraden }
   IF winkelhalbierende THEN
      BEGIN
         Plot (y, x, farbe);
         WHILE x <> ende DO
            BEGIN
               x := x + x1;
               IF e > 0 THEN
                  BEGIN
                     y := y + y1;
                     e := e -  b;
                  END
               ELSE
                  e := e + a;
               Plot (y, x, farbe);
            END
      END
   ELSE
      BEGIN

         Plot (x, y, farbe);
         WHILE x <> ende DO
            BEGIN
               x := x + x1;
               IF e > 0 THEN
                  BEGIN
                     y := y + y1;
                     e := e -  b;
                  END
               ELSE
                  e := e + a;
               Plot (x, y, farbe);
            END
      END;

END; { Prozedur 'Line' Ende }
```

Bild 6. Die vollständige Linie-Prozedur

ist dabei, die Berechnung auf INTEGER-Anweisungen zurückzuführen und möglichst nur die Grundrechenarten + und − zu verwenden. Ist dieses Ziel erreicht, wäre auch eine hardwaremäßige Lösung denkbar.
Schritt für Schritt wollen wir den oben gefundenen Algorithmus [2] optimieren:
Y: = ROUND (x * y2/x2)
Gemäß [2] muß gelten, daß der zu berechnende y-Wert gleich oder um eins größer ist als der vorherige. Bei Einhaltung der gefundenen Regeln ergibt sich das in *Bild 5* wiedergegebene Programm.

Eine hohe Verarbeitungsgeschwindigkeit ergibt sich aus drei Gründen:

- Die Anzahl der Operationen in der Schleife ist minimal.
- Es kann ausschließlich mit INTEGER-Variablen gearbeitet werden.
- Es treten (abgesehen von der Multiplikation mit 2, die sich durch ein „shift left" realisieren läßt) nur Addition und Subtraktion (keine zeitraubenden Punktrechnungen) auf.

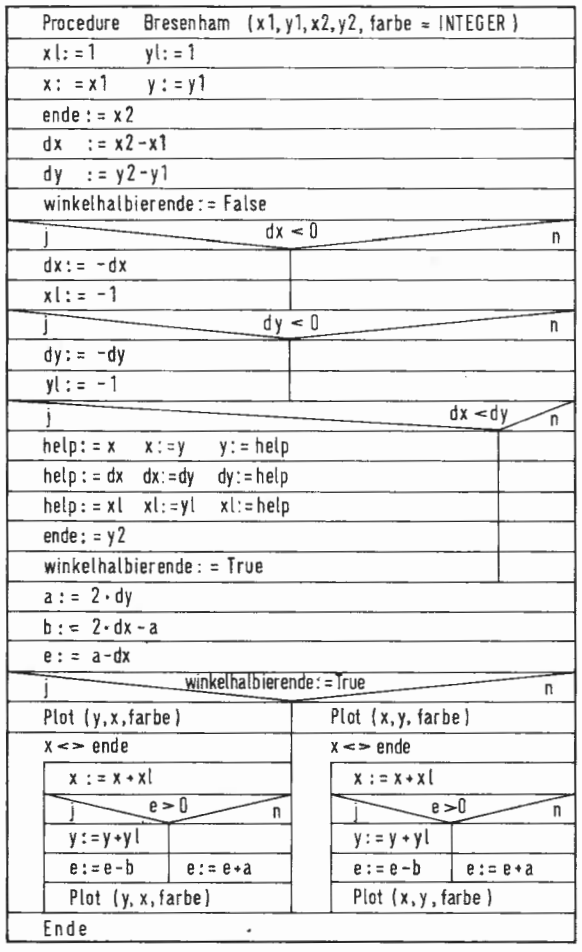


Bild 7. Der Programmablauf im Nassi-Shneidermann-Diagramm

```
Program Linetest;

{ Programm zur Einordnung der Ausführungsgeschwindigkeit des Bresenham-
{ Algorithmus
{ Benutzung des unter Turbo-Pascal implementierten Draw-Algorithmus

{ Der gesamte Bildschirm wird mit vertikalen Linien gefüllt.

VAR i       : INTEGER; { Schleifenvariable }
    Eingabe : Char;


BEGIN { Programm 'Linetest' Anfang }

   GraphMode;
   Palette (0);

   REPEAT UNTIL KeyPressed;
   READ (KBD, Eingabe);

   FOR i := 0 TO 319 DO
       Draw (i, 0, i, 199, 1);

   REPEAT UNTIL KeyPressed;
   READ (KBD, Eingabe);

   TextMode;

END.   { Programm 'Linetest' Ende   }
```

Bild 8. Der Bildschirm wird mit dem DRAW-Befehl gefüllt

Verallgemeinerung des Algorithmus

Um einen allgemeingültigen „Drawline-Algorithmus" für beliebige Geraden zu erhalten, versucht man alle möglichen Fälle des Linienverlaufes auf den besprochenen „Original-" Bresenham-Algorithmus zurückzuführen. Speziell müssen der beliebige Anfangspunkt sowie alle möglichen Steigungen der Geraden berücksichtigt werden.

a) Berücksichtigung des Anfangspunktes

Benutzt man einen beliebigen Anfangspunkt, so ergeben sich zwei Änderungen:
– Die Steigung der Geraden ist nicht mehr m = y2/x2. Statt dessen wird sie mit m = (y2−y1)/(x2−x1) berechnet. In diesem Sinne werden die beiden Variablen dx und dy definiert mit

dx: = (x2−x1)
dy: = (y2−y1)

Entsprechend sind in dem Algorithmus aus Bild 5 x2 durch dx und y2 durch dy zu ersetzen.

– Die Gerade soll erst ab dem Punkt P1 (x1,y1) gezeichnet werden. Dazu sind die Variablen (x,y), die den jeweils zu setzten Punkt bezeichnen, auf diesen Anfangswert zu setzen. Von diesem Punkt aus wird die Gerade dann, wie in Bild 5 gezeigt, gezeichnet.

b) Berücksichtigung der Steigung der Geraden

Mit Hilfe geeigneter Umrechnungen versucht man die möglichen Lagen der Gerade auf den Algorithmus aus Bild 5 zurückzuführen. Insgesamt sind vier Fälle zu unterscheiden:
– Die Gerade genügt den Restriktionen aus Bild 5, d. h. der „Original-Algorithmus" kann angewandt werden.

– Die Steigung der Geraden ist größer als m = 1. In diesem Fall muß eine Spiegelung an der Winkelhalbierenden erfolgen, d. h.

(x,y) → (y,x)

Der Algorithmus aus Bild 5 kann somit angewandt werden.
– Um den Originalalgorithmus verwenden zu können, muß die Gerade an der y-Achse gespiegelt werden, d. h. die Schleife aus Bild 5 muß in umgekehrter Richtung durchlaufen werden. (x → −x)
– Die Gerade muß an der y-Achse gespiegelt werden, um den Originalalgorithmus anzuwenden (y → −y). U. U. muß die Schleife in y-Richtung in entgegengesetzter Richtung durchlaufen werden (falls m ≤ −1).
Allgemein wird also bei Spiegelung an einer der Achsen in Abhängigkeit von der Steigung die Durchlaufrichtung der Schleife geändert, wobei gleichzeitig die Reihenfolge der Eingabe der Endpunkte P1 und P2 berücksichtigt wird.
Der in dieser Form verallgemeinerte Algorithmus ist in *Bild 6* in einer Pascal-Prozedur entwickelt worden (das Struktogramm zeigt *Bild 7*).
Zur Prüfung des Zeitverhaltens ist dieser Algorithmus auf einem Commodore PC10 unter Turbo-Pascal getestet worden.
Mit Hilfe der den *Bildern 8 und 9* zu entnehmenden Programme wurde der Bresenham-Algorithmus mit dem unter Turbo-Pascal vorhandenen DRAW-Algorithmus verglichen. Das Ausfüllen des Bildschirms mit senkrechten Linien dauerte mit dem implementierten Algorithmus ca. 27 Sekunden und mit dem in Pascal programmierten Bresenham-Algorithmus ca. 28 Sekunden. Für einen in Pascal programmierten Algorithmus eine ansehnliche Zeit.

```
Program Linetest;

{ Programm zur Einordnung der Ausführungsgeschwindigkeit des Bresenham-     }
{ Algorithmus                                                               }
{ Benutzung des programmierten Bresenham-Algorithmus                        }

{ Der gesamte Bildschirm wird mit vertikalen Linien gefüllt.                }

VAR i       : INTEGER; { Schleifenvariable }
    Eingabe : Char;

{$I Line.pas }   { Einbinden des Line-Algorithmus                           }


BEGIN { Programm 'Linetest' Anfang }

   GraphMode;
   Palette (0);

   REPEAT UNTIL KeyPressed;
   READ (KBD, Eingabe);

   FOR i := 0 TO 319 DO
       Line (i, 0, i, 199, 1);

   REPEAT UNTIL KeyPressed;
   READ (KBD, Eingabe);

   TextMode;

END.   { Programm 'Linetest' Ende   }
```

Bild 9. Mit der Prozedur Linie wird der Bildschirm gefüllt

Dipl.-Ing. Anton Obermaier

Sin und cosin mit Integer-arithmetik

Kleine Standalone-Systeme (wie zum Beispiel der Z80-EMUF) werden normalerweise nicht auf Fließkommarechnungen ausgelegt. Es gibt aber Methoden, nur mit Integerarithmetik Funktionen wie sin und cos im Rahmen solcher Minimalkonzepte mit hinreichender Genauigkeit zu implementieren.

Wenn man zum Beispiel den Z80 in einem kleinen Einplatinensystem einsetzt, dann muß man, will man nicht ein viel Speicherplatz fressendes Arithmetikpaket implementieren, schon für die gewöhnliche Multiplikation und Division von ganzen Zahlen eigene Routinen entwerfen. In der Praxis sollen aber oft auch höhere Funktionen von solchen Systemen berechnet werden. Bei geschickter Programmierung gelingt das auch – und solche Spezial-Routinen liefern ihr Ergebnis auch schneller ab, als allgemeine Gleitkommapakete das können. Die Schnelligkeit der Routinen und ihr geringer Speicherbedarf werden allerdings mit begrenzter Genauigkeit (hier vier Stellen) erkauft. In der Praxis ist aber oft sehr viel weniger Genauigkeit gefragt, als Gleitkommapakete sie liefern.
Als Beispiel soll angenommen werden, daß eine Positionierungsaufgabe im zweidimensionalen Raum gegeben ist, wobei Polarkoordinaten verwendet werden sollen. Hier wird die Sinus-Funktion mit Ausgabewerten zwischen – 1 und + 1 gebraucht, die zunächst nicht mit „Integers" berechenbar ist.

Zahlendarstellung

Trigonometrische Funktionen lassen sich mit Integer-Operationen programmieren, wenn man sich geeignete Datenstrukturen für die Argumente und die Zahlen aus dem Wertebereich der Funktion zurechtlegt. Jede der vorkommenden Größen ist in der Praxis beschränkt, es gibt einen Wert mit maximalem Betrag (wie dieser Wert für eine Variable aussieht, das ist je nach Einsatzfall verschieden, aber immer für eine solche Programmieraufgabe ermittelbar). Eine Lösung für die Darstellung solcher Größen verläuft so: Man nimmt die Integerzahl mit maximalem Betrag (sie sei mit Im bezeichnet), die man im gewählten Computersystem darstellen kann, ordnet ihr das Maximum m aus dem Wertebereich der Variablen zu und stellt alle kleineren Zahlen x aus dem Wertebereich im Computer dar, indem man diejenige Integerzahl Ix im Computer aufsucht, die sich aus der Formel

$$\frac{Ix}{I_m} = \frac{x}{m}$$ ergibt.

Also: $Ix = \frac{x\, I_m}{m}$,

wobei hier einfach später die Nachkommastellen weggelassen werden. Das ist so etwas wie „Normalisierung".
In Bezug auf Trigonometrie ergibt sich, wenn man mit 16-Bit-Worten umgeht (was beim Z80 naheliegend ist), nach diesem Schema eine natürliche Datenstruktur sowohl für Argumentwerte als auch für die Werte der trigonometrischen Funktionen:
1. Die Darstellung der Eingangsgröße (Argument = Winkel) als Wort liefert, wenn man die zugelassenen Werte auf 0...360 Grad beschränkt, eine Genauigkeit von 360/65536 Grad, was 19,8 Sekunden entspricht. Hier ist die Periodizität der trigonometrischen Funktionen sogar noch bei der Addition und Subtraktion mitberücksichtigt. Beispiel:

10 Grad wird dargestellt als die „Maschinenzahl", die dem Wert

(10/360)*65536 = 1820,4393

entspricht, also als 1820 dez = 071C H im Computer.

2. Die Werte von sin(x) und cos(x) müssen ebenfalls normalisiert dargestellt werden. Wenn hier 1 das Maximum m ist und zum Beispiel x = sin(10 Grad), dann ist

(sin(10 Grad)/1) =
0,000111000111010000.... bin,

was als

0001011000111010 bin = 163A H

gespeichert wird. m = 2 ↑ 15, weil cos(0°) = 1 dargestellt werden muß. Der relative Fehler beträgt dann −1,8 E-5.

Zur praktischen Berechnung genügt es, wenn jeder vorgegebene Winkel auf den ersten Quadranten reduziert wird. *Bild 1* zeigt, daß dort alle vorkommenden Werte positiv sind (0 =< x < PI/2 und 0 =< sin(x) , cos(x) =< 1). Die Algorithmen zur Berechnung des Betrages der beiden Funktionen gestalten sich hier besonders einfach. Die notwendigen Vorzeichen werden gesondert hinzugefügt.
Im vorliegenden Konzept werden nun sin und cos simultan berechnet. Das ist sinnvoll, weil in den meisten Fällen ohnehin beide Werte berechnet werden müssen und sowohl Code-Länge als auch Zeitverhalten in diesem Fall günstig sind.
Zum Beispiel dient die Formel

sin(x) = – cos(x-270Grad)

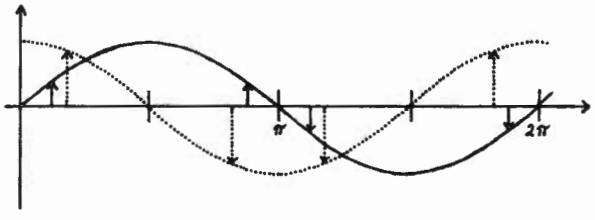


Bild 1. Alle Winkel lassen sich auf einen positiven Hauptbereich von 0 bis PI/2 reduzieren

Wertebereich	Maximum binär	Exponent zur Normalisierung
0 ≤ x ≤ 1,57...	1, ...	2^15
x^2 ≤ 2,46...	10, ...	2^14
x^4 ≤ 6,08...	110, ...	2^13
x^6 ≤ 15,0...	1111, ...	2^12
x^8 ≤ 37,0...	100101, ...	2^10

Bild 2. Die Faktoren der Normalisierung ergeben sich aus der Wertebeschränkung des Winkels x

als Ausgangspunkt für Berechnungen von sin, wenn x im 4. Quadranten liegt (anhand einer Tabelle werden alle vorkommenden Fälle später bearbeitet), woraus sich Reduktion und anschließend Vorzeichen und Vertauschung der Ergebnisse ableiten lassen. Tan(x) kann von hier aus mit einer einzigen Division errechnet werden. Die zur Berechnung noch benötigten weiteren Variablen werden in ähnlicher Weise normiert, wie schon geschehen.

Die theoretischen Grundlagen

Sin und cos kann man nach Lehrbuch in eine Reihe entwickeln:

$\sin(x) = x - (x \uparrow 3)/3! + (x \uparrow 5)/5! - (x \uparrow 7)/7! + \ldots$

Das kann man umformen in:

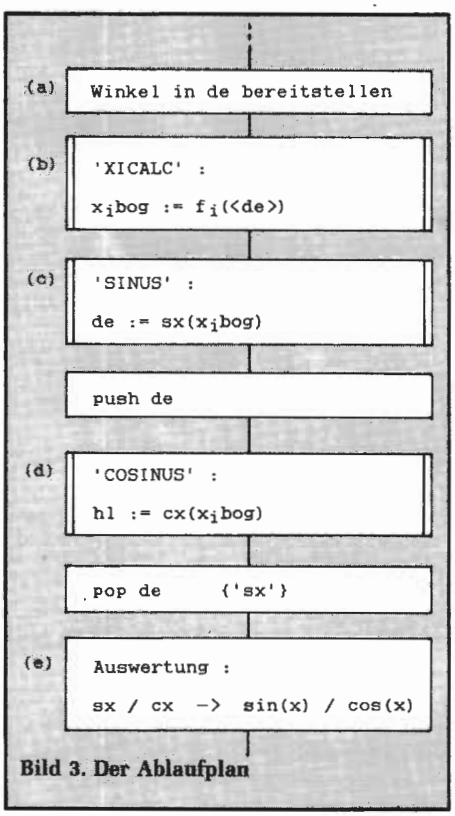


Bild 3. Der Ablaufplan

$\sin(x) = x * (1 - (x \uparrow 2)/3! + (x \uparrow 4)/5! - (x \uparrow 6)/7! + \ldots)$

während für cos gilt:

$\cos(x) = 1 - (x \uparrow 2)/2! + (x \uparrow 4)/4! - (x \uparrow 6)/6! + \ldots$

wobei x immer im Bogenmaß anzugeben ist. In den letzten beiden Formeln taucht x immer mit geraden Exponenten auf, was die gemeinsame Berechnung von sin und cos sehr vereinfacht.
Die Genauigkeit bei den einzelnen Gliedern hängt in erster Linie von der verwendeten Stellenzahl bei der Quadrierung ab. Normalisiert man hier wie vorhin geschildert, ergeben sich die in *Bild 2*

```
                ;
                ; Programmodul 'XICALC' zur Aufbereitung von Daten
                ; fuer die anschliessende Berechnung von sin(x) / cos(x)
                ;
                ; IN    reg de : Winkel x
                ;
                ; OUT   var x1bog : Winkel in Bogenmass
                ;       var x2bog : x1bog^2
                ;       var x4bog : x1bog^4
                ;       var x6bog : x1bog^6
                ;       var x8bog : x1bog^8
                ;
                ; REG   a,f,b,c,d,e,h,l
                ;

                        extrn multi

2000 =          ram     equ 2000h       ; Festlegung des Variablenblocks
2000 =          x1bog   equ ram         ;   im RAM
2002 =          x2bog   equ x1bog + 2
2004 =          x4bog   equ x2bog + 2
2006 =          x6bog   equ x4bog + 2
2008 =          x8bog   equ x6bog + 2
C910 =          pi2     equ 51472       ; Konstante pi2 := PI * 2^14

0000 7A                 ld a,d          ; Reduktion des Winkels x auf
0001 E63F               and 00111111b   ;   0 <= x < 90  [Grad]
0003 57                 ld d,a

0004 0110C9             ld bc,pi2       ; Umrechnung von x [Grad]
0007 CD0000             call multi      ;   in x1bog [Bogenmass]
000A CB14               rl h            ; Division <de>:=<dehl>/2^14
000C CB13               rl e
000E CB12               rl d
0010 CB14               rl h
0012 CB13               rl e
0014 CB12               rl d
0016 ED530020           ld (x1bog),de   ; Ergebnis in x1bog abspeichern

001A 42                 ld b,d          ; x1bog quadrieren
001B 4B                 ld c,e
001C CD0000             call multi
001F ED530220           ld (x2bog),de   ; Ergebnis in x2bog abspeichern

0023 42                 ld b,d          ; x2bog quadrieren
0024 4B                 ld c,e
0025 CD0000             call multi
0028 CB14               rl h            ; Division <de>:=<dehl>/2^15
002A CB13               rl e
002C CB12               rl d
002E ED530420           ld (x4bog),de   ; Ergebnis in x4bog abspeichern

0032 CD0000             call multi      ; x2bog * x4bog berechnen
0035 CB14               rl h            ; Division <de>:=<dehl>/2^15
0037 CB13               rl e
0039 CB12               rl d
003B ED530620           ld (x6bog),de   ; Ergebnis in x6bog abspeichern

003F CD0000             call multi      ; x2bog * x6bog berechnen
0042 ED530820           ld (x8bog),de   ; Ergebnis in x8bog abspeichern

                        ; alle xibog vorbereitet

0046 C9                 ret
```

Bild 4. XICALC bereitet vor

```
                ;
                ; Programmodul 'SINUS' zur Berechnung von sin(x)
                ; nach Vorbereitung der Daten durch Modul 'XICALC'
                ;
                ; IN    var x1bog : Winkel in Bogenmass
                ;       var x2bog : x1bog^2
                ;       var x4bog : x1bog^4
                ;       var x6bog : x1bog^6
                ;       var x8bog : x1bog^8
                ;
                ; OUT   reg de : var sx = sin(x) * 2^15
                ;
                ; REG   a,f,b,c,d,e,h,l
                ;

                        extrn multi,divi

2000 =          ram     equ 2000h       ; Festlegung des Variablenblocks
2000 =          x1bog   equ ram         ;   im RAM
2002 =          x2bog   equ x1bog + 2
2004 =          x4bog   equ x2bog + 2
2006 =          x6bog   equ x4bog + 2
2008 =          x8bog   equ x6bog + 2

8000 =          fs1     equ 8000h       ; Koeffizienten fsi der i-ten Glieder
0003 =          fs2     equ 3
001E =          fs3     equ 2*3*5
0276 =          fs4     equ 3*5*6*7
2C4C =          fs5     equ 2*3*5*6*7*9

0000 ED5B0220           ld de,(x2bog)   ; Berechnung des 2. Gliedes
0004 010300             ld bc,fs2       ;   der Reihe
0007 CD0000             call divi
000A 110080             ld de,fs1       ; erste Zwischensumme berechnen
000D EB                 ex de,hl        ;   und zwischenspeichern
000E AF                 xor a
000F ED52               sbc hl,de
0011 E5                 push hl

0012 ED530420           ld de,(x4bog)   ; Berechnung des 3. Gliedes
0016 011E00             ld bc,fs3
0019 CD0000             call divi

001C D1                 pop de          ; Zwischensumme im Stack
001D 19                 add hl,de       ;   aktualisieren
001E E5                 push hl

001F ED530620           ld de,(x6bog)   ; Berechnung des 4. Gliedes
0023 017602             ld bc,fs4
0026 CD0000             call divi

0029 EB                 ex de,hl        ; Zwischensumme im Stack
002A E1                 pop hl          ;   aktualisieren
002B AF                 xor a
002C ED52               sbc hl,de
002E E5                 push hl

002F ED530820           ld de,(x8bog)   ; Berechnung des 5. Gliedes
0033 014C2C             ld bc,fs5
0036 CD0000             call divi
```

Bild 5. Das Programm SINUS

```
0039 D1                 pop de          ; Stack korrigieren
003A 19                 add hl,de       ;   und Summe fertigstellen

003B EB                 ex de,hl        ; Summe mit Vorfaktor multiplizieren
003C ED4B0020           ld bc,(x1bog)
0040 CD0000             call multi

0043 CB14               rl h            ; Division <de>:=<dehl>/2^15
0045 CB13               rl e
0047 CB12               rl d

                        ; sx in reg de berechnet

0049 C9                 ret
```

```
                ;
                ; Programmodul 'COSINUS' zur Berechnung von cos(x)
                ; nach Vorbereitung der Daten durch Modul 'XICALC'
                ;
                ; IN    var x1bog : Winkel in Bogenmass
                ;       var x2bog : x1bog^2
                ;       var x4bog : x1bog^4
                ;       var x6bog : x1bog^6
                ;       var x8bog : x1bog^8
                ;
                ; OUT   reg hl : var cx = cos(x) * 2^15
                ;
                ; REG   a,f,b,c,d,e,h,l
                ;

                        extrn divi

2000 =          ram     equ 2000h       ; Festlegung des Variablenblocks
2000 =          x1bog   equ ram         ;   im RAM
2002 =          x2bog   equ x1bog + 2
2004 =          x4bog   equ x2bog + 2
2006 =          x6bog   equ x4bog + 2
2008 =          x8bog   equ x6bog + 2

8000 =          fc1     equ 8000h       ; Koeffizienten fci der i-ten Glieder
0006 =          fc3     equ 2*3
005A =          fc4     equ 3*5*6
04EC =          fc5     equ 2*3*5*6*7

0000 210080             ld hl,fc1       ; erste Zwischensumme berechnen
0003 ED530220           ld de,(x2bog)   ;   und auf Stack ablegen
0007 AF                 xor a
0008 ED52               sbc hl,de
000A E5                 push hl

000B ED530420           ld de,(x4bog)   ; Berechnung des 3. Gliedes
000F 010600             ld bc,fc3
0012 CD0000             call divi
```

Bild 6. COSINUS

```
0015 D1              pop de          ; Zwischensumme im Stack
0016 19              add hl,de       ;   aktualisieren
0017 E5              push hl

0018 ED530620        ld de,(x6bog)   ; Berechnung des 4. Gliedes
001C 015A00          ld bc,fc4
001F CD0000          call divi

0022 D1              pop de          ; Zwischensumme im Stack
0023 EB              ex hl,de        ;   aktualisieren
0024 AF              xor a
0025 ED52            sbc hl,de
0027 E5              push hl

0028 ED530820        ld de,(x8bog)   ; Berechnung des 5. Gliedes
002C 01EC04          ld bc,fc5
002F CD0000          call divi

0032 D1              pop de          ; Stack korrigieren
0033 19              add hl,de       ;   und Summe fertigstellen

                     ; cx in reg hl berechnet

0034 C9              ret
```

Quadrant	Bit 7 / 6	2^15*sin(x)	2^15*cos(x)
1	0 0	<de>	<hl>
2	0 1	+ <hl>	- <de>
3	1 0	- <de>	- <hl>
4	1 1	- <hl>	+ <de>

Bild 7. Die Zuordnung der Ergebniswerte zum Argumentwert

```
;
; Programmodul 'MULTI' zur Multiplikation
; von zwei 16 Bit Zahlen
;
; IN   reg bc : Multiplikator
;      reg de : Multiplikand
;
; OUT  reg bc : Multiplikator (bleibt erhalten)
;      reg de : hoeherwertige 16 Bit des Produkts
;      reg hl : niederwertige        "
;
; REG  a,f,d,e,h,l
;
```

Bild 8. Die Programme für Multiplikation und Division

```
             multi:
0000 3E10            ld a,16 .       ; Zaehler vorbelegen
0002 210000          ld hl,0

0005 29      mloop   add hl,hl       ; <dehl> links schieben und
0006 CB13            rl e            ;   hochwertiges Bit
0008 CB12            rl d            ;   ueberpruefen
000A 3004            jr nc,nosum

000C 09              add hl,bc       ; aufaddieren
000D 3001            jr nc,nosum

000F 13              inc de          ; Uebertrag in hochwertiges
                                     ;   Wort eintragen
0010 3D      nosum   dec a
0011 20F2            jr nz,mloop     ; fuer jedes Bit in de
                                     ;   wiederholen

                     ; Produkt in dehl

0013 C9              ret

                     ;
                     ; Programmodul 'DIVI' zur Division von
                     ; zwei 16 Bit Zahlen
                     ;
                     ; IN   reg bc : Divisor
                     ;      reg de : Dividend
                     ;
                     ; OUT  reg bc : Divisor (bleibt erhalten)
                     ;      reg de : Rest der Division
                     ;      reg hl : Quotient
                     ;
                     ; REG  a,f,d,e,h,l
                     ;
             divi:
0014 AF              xor a           ; Zaehler vorbelegen
0015 67              ld h,a
0016 6F              ld l,a
0017 3E10            ld a,16

0019 CB13    dloop   rl e            ; <de> links schieben
001B CB12            rl d
001D ED6A            adc hl,hl       ; <hl> links schieben
001F ED42            sbc hl,bc       ; Subtraktion
0021 3001            jr nc,diffok

0023 09              add hl,bc       ; Sub. revidieren
0024 3F      diffok  ccf

0025 3D              dec a
0026 20F1            jr nz,dloop     ; fuer jedes Bit in de
                                     ;   wiederholen

                     ; Quotient in hl, Rest in de

0028 C9              ret
```

gezeigten Faktoren. Diese Variablen seien im folgenden mit xibog bezeichnet, wobei i die Potenz bei x angebe. Die Fehler bei den xibog sind dann recht gering, wenn sie nach Bild 2 dargestellt werden. Der Gesamtfehler einer Berechnung von sin und cos mit Reihenentwicklung hängt dann im wesentlichen vom Abbruchkriterium während der iterativen Berechnung der Reihe ab.
Da das Glied mit x ↑ 10 in beiden Reihen oben wegen des starken Anwachsens der Fakultät bei kleinen x nur noch Beiträge von der Größenordnung 2,5 E-5 (beim sin sogar 2,3 E-6) liefert, werden alle Summanden ab einschließlich x10bog nicht mehr berücksichtigt.

Mit den Variablen

x1bog = x * 2 ↑ 15
x2bog = x ↑ 2 * 2 ↑ 14
x4bog = x ↑ 4 * 2 ↑ 13
x6bog = x ↑ 6 * 2 ↑ 12
x8bog = x ↑ 8 * 2 ↑ 10

ist

sx :=
(1/(2 ↑ 15)) * ((x1bog) * ((2 ↑ 15) – (x2bog)/3 + (x4bog)/(2*3*5) – (x6bog)/(3*5*6*7) + (x8bog)/(2*3*5*6*7*9)))
≈ sin(x) * 2 ↑ 15

und

cx := (2 ↑ 15) – (x2bog) + (x4bog)/(2*3) – (x6bog)/(3*5*6) + (x8bog)/(2*3*5*6*7) ≈ cos(x) * 2 ↑ 15

Diese Formeln sind in den Listings realisiert. Dabei sind die Klammern so gesetzt, daß bei der Umsetzung in das Programm die Genauigkeit erhalten bleibt.

Die Implementierung

Entsprechend den drei Teilproblemen gibt es drei Module, XICALC, SINUS und COSINUS, die linear aufgebaut sind. Dazu kommen noch einige Hilfs-Routinen, die etwas Rechenarbeit leisten. *Bild 3* zeigt einen Ablaufplan. Im Z80-Registerpaar DE wird der Winkel erwartet, wobei die Normalisierung wie besprochen durchgeführt sein soll. XICALC berechnet daraus die fünf Variablen xibog. Die Reduzierung der Argumente auf den ersten Quadranten besteht hier aufgrund der Darstellung der Argumentwerte in der Ausmaskierung der beiden höchstwertigen Bits des Winkels, also der Bits 7 und 6 in Register D des Z80. Der Winkel x wird mit

Spruch des Monats

„Im Gegensatz zu vielen anderen Bereichen haben auf dem Computersektor die Männer keinen Vorsprung vor den Frauen."

Renate Kelava-Lendzian, Geschäftsführerin der SPSS in München bei der Tagung „Die Sekretärin im Management der Zukunft"

x [Grad] = <DE> * 360 / 2 ↑ 16
in
x1bog = x * 2 ↑ 15 = <DE> * PI
= (<DE> * (PI * 2 ↑ 14))/2 ↑ 14

umgewandelt, wobei auch PI normalisiert angegeben sein muß.

Die Divisionen im Programm sind soweit als möglich auf Rotationen der Register zurückgeführt. Die Unterprogramme MULTI und DIVI halten Parameterübergabebedingungen ein, die alles sehr schnell machen. Das Ergebnis der Multiplikation beim Umrechnen nach den Formeln oben steht in DE–HL (32 Bit). Division durch 2 ↑ 14 bedeutet Stellenverschiebung um 14 Positionen nach rechts, was auch so aufgefaßt werden kann, daß die Bits 14 bis 29 aus DE–HL in ein 16 Bit breites Register geladen werden (die höchstwertigen beiden Bits werden wegmaskiert). Mit Linksverschiebung um zwei Stellen steht das Ergebnis in DE. *Bild 4* zeigt das Listing. *Bild 5* zeigt die Routine SINUS, die die Werte xibog aus dem RAM nimmt und dann sx in DE ausgibt. Das Ergebnis sx muß vor dem Aufruf von COSINUS zwischengespeichert werden, was einfach wegen der einheitlichen Struktur der Module nicht automatisch von SINUS gemacht wird.
Bild 6 zeigt die Routine COSINUS, die ebenfalls auf die xibog zurückgreift. Das Ergebnis cx wird in HL geliefert. Nachdem sx wieder in DE installiert ist, stehen beide Werte in den Z80-Registern zur Verfügung.

Die Auswertung

Bei der Reduktion auf den ersten Quadranten müssen vier Fälle beachtet werden, die den vier Quadranten entsprechen und die aus den bei x wegmaskierten beiden höchstwertigen Bits abgeleitet werden können. *Bild 7* zeigt tabellarisch, wie sx und cx bezüglich Vorzeichen und Zuordnung zu sin und cos interpretiert werden müssen. Das Beispiel am Anfang mit sin(x) = – cos (x – 270 Grad) könnte verbal so interpretiert werden: Der Sinus eines Winkels im 4. Quadranten ist mit dem negativen Cosinus des auf den 1. Quadranten reduzierten Winkels identisch. Sin(x) ist also in solch einem Fall der Wert cx in HL, der noch dazu mit dem negativen Vorzeichen versehen werden muß.

Was bringt es an Effizienz?

Tests mit den hier abgebildeten Programmen haben gezeigt, daß Sinus und Cosinus bei 4 MHz Taktfrequenz in 4 ms berechnet werden können. Eine Gleitkomma-Rechnung auf dem 68000 nach dem Gleitkommapaket aus mc, Ausgabe 2/1985, dauerte 19 mal so lang (allerdings bei doppelter Zahl gültiger Stellen). Das Beispiel mit sin(10 Grad) liefert als Ergebnis

<DE> = 1638 H = 5688 dez.

Umgerechnet ist das 0,17358398 ..., was einen relativen Fehler von -3.7 E-4 bedeutet. Der mitberechnete Cosinus lag mit -2,5 E-5 Abweichung noch genauer. Die Länge der Quelle ist mit 200 Byte noch recht handlich. Die verwendeten externen Module zur Multiplikation und Division von 16-Bit-Zahlen sind ohnehin notwendig, müssen aber bestimmten Ein- und Ausgabebedingungen genügen. *Bild 8* zeigt mögliche Varianten mit den vorgeschriebenen Merkmalen.

Literatur

[1] *Bronstein – Semendjajew:* Taschenbuch der Mathematik. Verlag Harri Deutsch.
[2] *Lohninger, H:* Gleikomma-Arithmetik für den 68000. mc, Ausgabe 2, 1985.

Helmut Winkelmann

Menüauswahl

Turbo-Pascal-Programme mit mehr Komfort

Die Zeiten, in denen an ein Programm nur die Anforderung gestellt wurde, daß es korrekt arbeitet, sind vorbei. Andere Forderungen wie z.B. Wartbarkeit und Benutzerfreundlichkeit sind hinzugekommen. Unser Autor hat ein Turbo-Pascal-Programm geschrieben, das die Auswahl von Menüpunkten mit den Cursortasten ermöglicht.

Bei der Programmentwicklung lautet die Forderung immer häufiger: „Interaktive Auswahl unter mehreren Programmpunkten". Ein Beispiel dafür ist die Darstellung aller Dateinamen einer Diskette auf dem Bildschirm und die Auswahl einer Datei zur Bearbeitung. Eine sehr benutzerfreundliche Methode der Auswahl gestattet das Unterprogramm „Auswahl" (*Bild*), geschrieben in Turbo-Pascal für IBM-PC und Kompatible.

Man stelle sich vor, daß ein bestimmter Bildschirmbereich als Fenster definiert wurde (z.B. mit procedure window()) und daß das Fenster in Spalten und Zeilen aufgeteilt ist, ähnlich einem Spreadsheet in der Tabellenkalkulation. Jedes Feld wird jetzt mit einem Text zur Auswahl belegt. Ein bestimmtes Feld wird z.B. hell oder in einer anderen Farbe optisch hervorgehoben. Anschließend kann man mit den Cursortasten andere Felder anfahren, bzw. optisch hervorheben, und, falls man sich für eines entschieden hat, durch Betätigen der Enter-Taste auswählen. Die getroffene Auswahl wird dann ans aufrufende Programm übergeben. Als Eingabeparameter der Funktion „Auswahl" braucht man ein „array of string" mit dessen Elementen die Felder auf dem Bildschirm belegt werden, sowie die Anzahl der Elemente. Als Rückgabe der Funktion „Auswahl" bekommt man den Index des ausgewählten Elementes im „array of string".

Wie erfolgt nun die Organisation des Fensters? Um nicht nur die Anwendung der Funktion „Auswahl" zu erleichtern, sondern auch ihre Programmierung, ermittelt „Auswahl" fast alles selbst. Im Hauptprogramm muß lediglich das Feld mit den Auswahltexten definiert werden, sowie eine Konstante, die die maximale Länge eines Auswahltextes festlegt (txtlen, maxnum). Je nach Länge der Texteinträge ergibt sich die Spaltenzahl (maxspalte), wobei als Trennzeichen hier ein Blank verwendet wird. Aus Spaltenzahl und Anzahl der Einträge überhaupt ergibt sich die Zeilenzahl (maxzeile). Es muß nicht jede Zeile voll belegt sein, verbotene Felder (leer) können nicht ausgewählt werden.

Der Programmablauf

Nach Aufruf von „Auswahl" wird das definierte Fenster gelöscht und die aktuelle Zahl der Spalten und Zeilen ermittelt, die benötigt werden. Dabei ist zu beachten, daß Zeilen, Spalten und Feldindizes von 0 ab gerechnet werden, die Anzahl der übergebenen Feldwerte aber bei 1 beginnt (logischerweise). Anschließend werden alle Auswahltexte auf ihrem zugeordneten Feld dargestellt. An die letzte Spalte in jeder Zeile wird dabei kein Blank (Trennzeichen) angehängt, da dies erstens nicht notwendig ist und zweitens wird damit vermieden, daß der Teletypeschreibstil von Turbo-Pascal in der letzten Zeile des Fensters eine Zeilenschaltung bewirkt und damit das ganze Fenster verschiebt.

Als aktuelles Element wird das Element 0, also das Feld mit Zeile 0 und Spalte 0

```
program auswahlbeispiel;

                    { ***** Allgemeine Prozeduren ***** }

type
  somechr = set of char;
function enter(group:somechr):char;
var chr : char;
begin
  repeat
    read(kbd,chr);
    chr := upcase(chr);
  until chr in group;
  enter := chr;
end;  {enter}

                        { *****  Funktion Auswahl  ***** }

const
  txtlen = 15;
  maxnum = 99;
type
  textfeld = array [0..maxnum] of string[txtlen];      { 0..99 : 100 Eintraege }

function auswahl(var auswtext : textfeld; maxwahl:byte):byte;
type
  postype = record
              oldz,oldsp,newz,newsp : byte;
            end;
var
  pos : postype;                    { Zwischenspeicher fuer neue,alte Zeile,Spalte }
  maxzeile,maxspalte,               { aktuelle max-Werte fuer Zeile, Spalte }
  z,s,
  spaltbr,                          { Breite einer Textspalte }
  altewahl,neuewahl : byte;         { Index ergibt sich aus oldz,oldsp... }
  ende : boolean;
  zeichen : char;
```

Mit diesen Prozeduren werden selbstgeschriebene Programme durch Menüauswahl einfacher bedienbar

```
  procedure highlight(var pos:postype);
  begin
    with pos do
    begin
      gotoxy(oldsp * spaltbr +1,oldz +1);        { gehe zu alter Position }
      lowvideo;                                  { altes Feld dunkeltasten }
      write(auswtext[altewahl]:txtlen);
      if oldsp < maxspalte then write(' ');      { Trennzeichen }
      gotoxy(newsp * spaltbr +1,newz +1);        { gehe zu neuer Position }
      highvideo;                                 { neue Auswahl helltasten }
      write(auswtext[neuewahl]:txtlen);
      if newsp < maxspalte then write(' ');
      lowvideo;                                  { wieder normaler Videomodus }
      oldz := newz; oldsp := newsp;              { neue Werte merken }
      altewahl := neuewahl;
    end;
  end;

begin { auswahl }
  window(1,1,80,20);                          { z.B. Fenster fuer Auswahlfeld }
  clrscr; lowvideo;
  spaltbr := txtlen +1;                { wegen Leerzeichen zwischen Spalten }
  maxspalte := 80 div spaltbr;         { wieviele Spalten passen ins Fenster }
  if maxwahl < maxspalte
  then maxspalte := maxwahl-1          { nur eine Auswahlzeile }
  else maxspalte := maxspalte-1;       { mehrere Zeilen mit Spalte 0..4 }
  maxzeile := (maxwahl-1) div (maxspalte+1);                  { 0..n }
  for z := 0 to maxzeile do
  begin                                { alle Felder mit Text belegen }
    gotoxy(1,z+1);
    for s := 0 to maxspalte do
    begin
      write(auswtext[z* (maxspalte+1) +s]:txtlen);
      if s < maxspalte then write(' ');{ Trennzeichen nur zwischen Spalten }
    end;
  end;

  with pos do
  begin
    oldz := 0; oldsp := 0;             { Auswahl auf erstes Element (#0) setzen }
    newz := 0; newsp := 0;
    altewahl := 0; neuewahl := 0;
    highlight(pos);
    ende := false;
    repeat
      zeichen := enter([^M,^[]);       { Tastaturabfrage beschraenken }
      case zeichen of
      ^M: ende:=true;                  { Return = Auswahl beendet }
      ^[: begin
            read(kbd,zeichen);         { 2.Zeichen des Steuercodes lesen }
            case zeichen of
            'H': if oldz > 0 then newz := pred(oldz);          { Zeile oben }
            'P': if oldz < maxzeile then newz := succ(newz);   { Zeile unten }
            'K': if oldsp > 0 then newsp := pred(oldsp);       { Spalte links }
            'M': if oldsp < maxspalte then newsp := succ(newsp);{Spalte rechts}
            end;  {case zeichen of }
            neuewahl := newz * (maxspalte+1) + newsp;          { neuer Index }
            if neuewahl > pred(maxwahl)
            then
              begin                    { keine Aenderung, da Feld nicht belegt }
                newsp := oldsp; newz := oldz; neuewahl := altewahl;
              end;
            highlight(pos);
          end;
      end;       {case}
    until ende;
    auswahl := neuewahl;
  end;     { with }
end;  {auswahl}

                    { ***** Testprogramm ***** }

var
  testfeld : textfeld;
  i : integer;
begin
  clrscr;
  gotoxy(1,23);
  write('Bitte mit Cursortasten Feld anwaehlen und mit Enter auswaehlen !');
  for i := 0 to 17 do testfeld[i] := char(i+65)+char(i+65); { Feld laden }
  for i := 18 to 99 do testfeld[i] := '';                   { Rest loeschen }
  i := auswahl(testfeld,18);                                { do it }
  window(1,1,80,25);
  gotoxy(1,24);
  write('Auswahl (Feldindex) : ',i,',      Feldinhalt : ',testfeld[i]);
end.    { of testprogram }
```

definiert. Eine Unterprocedure (highlight) besorgt die optische Hervorhebung dieses Feldes. Der Rest von „Auswahl" besteht aus einer Schleife, die ständig die Tastatur abfragt. Falls eine Cursortaste gedrückt wird, werden die neuen Positionen sowie der neue Index ermittelt (falls erlaubt) und die Helltastung wird entsprechend gesetzt. Wird die Enter-Taste gedrückt, wird die Schleife abgebrochen und der aktuelle Index zurückgegeben.

Der Übersichtlichkeit halber sind die Positionsvariablen (oldz, oldsp, newz, newsp) in einem Record „pos" zusammengefaßt.

Die Prozedur „Highlight"

Die Unterprozedur „highlight" sorgt für die optische Kennzeichnung der angewählten Felder. Dies geschieht folgendermaßen:

- Berechnen der Cursorposition für das alte Feld (noch angewählt)
- Überschreiben in der „Normalform",
- Berechnen der Cursorposition für das neu angewählte Feld,
- Überschreiben in der „Auswahlform",
- Variablen zur Speicherung der alten Position neu setzen.

Für die besondere Darstellung wurde hier nur „Helltasten" benutzt, da dies mit fast allen Monitoren funktioniert. Je nach Ausstattung des PCs können hier auch andere Verfahren benutzt werden, wie z.B. inverse Darstellung, andere Farben etc.

Die Funktion „Enter"

Interessant ist noch die sehr praktische Funktion „enter", die der Autor in jedem seiner Programme benutzt. An diese Funktion wird ein „Set of char" übergeben und „enter" frägt dann die Tastatur solange ab, bis eines der vorgegebenen Zeichen eingegeben wird. Das eingegebene Zeichen wird dann als Ergebnis zurückgeliefert. Eine automatische Umwandlung in Großbuchstaben erfolgt, damit bei der Auswahlvorgabe nicht große und kleine Buchstaben angegeben werden müßten (Schreibarbeit).

Das Testprogramm führt die Funktion „Auswahl" vor und bedarf wohl keiner näheren Erläuterung.

Luidger Röckrath

GEM für alle PCs

Die grafische Bedieneroberfläche GEM, angelehnt an das Betriebssystem des MacIntosh von Apple, hat eine große Zahl begeisterter Anwender gefunden. Man kann es auf dem Atari 520-ST einsetzen, auf dem Original-PC/XT/AT von IBM und auf den Compaq-Modellen. Leider hört GEM auf anderen IBM-Kompatiblen schon beim Booten mit der Arbeit auf. mc bringt ein paar kleine Änderungen, mit welchen GEM auf allen Kompatiblen läuft.

Der Fall ist realistisch: Ein Besitzer eines Kompatiblen hält GEM für eine schöne Sache und bestellt es bei seinem Distributor, der es ausdrücklich als für den Olivetti M24 geeignet empfiehlt. Das Programm wird gekauft und sofort nach Anleitung installiert. Nach dem Start erscheinen aber nicht die schönen Fenster, Ikons, Pull-Down-Menues und alle anderen Wunderdinge moderner Bedienerführung, sondern es erscheint nur die lapidare Meldung:

Diese Version von GEM läuft nicht auf diesem Computer.

Dann meldet sich das Betriebssystem mit A>. Ein zweiter Versuch bringt den gleichen Effekt. Das alles auf einem Modell, auf dem von Sidekick bis zum MS-Flugsimulator alles Kritische einwandfrei läuft.

So geschehen vor einiger Zeit auf meinem Olivetti-PC. Die Ausstiegsmeldung kam damals zwar in englischer Sprache, aber so hat es sich zugetragen. Ich wollte mich damit nicht zufrieden geben, denn aus der Pionierzeit der Mikrocomputer war mir der Umgang mit Bit und Byte (auf dem TRS-80) gut vertraut, und ich konnte nicht glauben, daß da nichts zu machen sein würde.

```
C>debug gemvdi.exe
-r
AX=0000  BX=0000  CX=1E00  DX=0000  SP=02EE  BP=0000  SI=0000  DI=0000
DS=2AC9  ES=2AC9  SS=2C71  CS=2AD9  IP=11E1   NV UP DI PL NZ NA PO NC
2AD9:11E1 2E            CS:
2AD9:11E2 8C06000D      MOV     [0D00],ES                       CS:0D00-000C
-rds
DS 2AC9
:2ad9
-s011e00,'Diese Version von GEM läuft nicht auf diesem Computer'
2AD9:16CA
-d16ca,16ff
2AD9:16C0                                       44 69 65 73 65 20          Diese
2AD9:16D0  56 65 72 73 69 6F 6E 20-76 6F 6E 20 47 45 4D 20   Version von GEM
2AD9:16E0  6C 84 75 66 74 20 6E 69-63 68 74 20 61 75 66 20   l.uft nicht auf
2AD9:16F0  64 69 65 73 65 6D 20 43-6F 6D 70 75 74 65 72 24   diesem Computer$
-s011e00,ca,16
2AD9:1205
-u11e6,120c
2AD9:11E6 1E            PUSH    DS
2AD9:11E7 BBEFF0        MOV     BX,F0EF
2AD9:11EA 8EDB          MOV     DS,BX
2AD9:11EC 817F0B434F    CMP     WORD PTR [BX+0B],4F43
2AD9:11F1 7417          JZ      120A
2AD9:11F3 813E1ED14942  CMP     WORD PTR [D11E],4249
2AD9:11F9 740F          JZ      120A
2AD9:11FB 813E33D1436F  CMP     WORD PTR [D133],6F43
2AD9:1201 7407          JZ      120A
2AD9:1203 1F            POP     DS
2AD9:1204 BACA16        MOV     DX,16CA
2AD9:1207 E9F3FE        JMP     10FD
2AD9:120A B419          MOV     AH,19
2AD9:120C CD21          INT     21
-u10fd,111a
2AD9:10FD 0E            PUSH    CS
2AD9:10FE 1F            POP     DS
2AD9:10FF B409          MOV     AH,09
2AD9:1101 CD21          INT     21
2AD9:1103 BA5F0D        MOV     DX,0D5F
2AD9:1106 B409          MOV     AH,09
2AD9:1108 CD21          INT     21
2AD9:110A BAD92A        MOV     DX,2AD9
2AD9:110D 8EDA          MOV     DS,DX
2AD9:110F BA8011        MOV     DX,1180
2AD9:1112 B43B          MOV     AH,3B
2AD9:1114 CD21          INT     21
2AD9:1116 32C0          XOR     AL,AL
2AD9:1118 B44C          MOV     AH,4C
2AD9:111A CD21          INT     21
-q
```

Bild 1. Mit Debug kann man raffiniert ändern

Es läuft doch

Zu MS-DOS wird der komfortable Monitor DEBUG mitgeliefert. Den habe ich eingesetzt, um GEM zu untersuchen. Die Ursache des Ausstiegs konnte damit schnell eingekreist und ausgemerzt werden. Das Programm GEM ist nämlich doch auf dem Olivetti M24 (und jedem anderen Kompatiblen) genauso lauffähig wie auf dem Original, hätte Digital Research nicht bewußt versucht, dies zu verhindern: Es wird beim Start der Copy-Right-Vermerk abgefragt, und wenn dort nicht zwei der drei magischen Buchstaben stehen, die zum Glück noch nicht die ganze Computerwelt beherrschen, dann beendet das Programm flink seine Arbeit, ohne überhaupt richtig etwas getan zu haben. Was man dagegen tun kann: Abfrage entfernen und Programm neu starten, das ist alles!

Man fragt sich natürlich, weshalb Digital Research nur ein Drittel des Marktes beliefern wollte, was als Marketing-Strategie doch problematisch zu sein scheint. Inzwischen wurde klar, daß Digital Research GEM für jeden Kompatiblen ausliefern wollte, aber in einer speziell angepaßten Version. Welche Hersteller diesbezüglich OEM-Verträge mit Digital Research abgeschlossen haben, ist nicht ganz klar. Neben der IBM-Version gibt es auch eine für den Siemens PC-D. Eine lang angekündigte Version für Olivetti mit Treiber für die hochauflösende Grafik von 640 * 400 Pixel wurde schon auf Messen gesichtet, scheint aber noch nicht im Handel zu sein. Hätte Digital Research seinen Schutzmechanismus nicht so primitiv gestaltet, dann wäre GEM wohl den Besitzern von xy-Taiwan-Import-PCs versagt geblieben.

100 % kompatibel?

Die sture Abeitsverweigerung von GEM werden auch viele Besitzer sogenannter 100%-Kompatibler erfahren haben. Wie steht es also wirklich um diesen eher schillernden Begriff?

Kompatibilität heißt funktionelle

Gleichheit, unabhängig von der internen Realisierung. Nur so kann die Entwicklung kompatibler Rechnersysteme rechtlich unanfechtbar bleiben. Auf der Hardware-Seite greifen rechtliche Schutzmechanismen nicht recht, hier heißt IBM-Kompatibilität oft Identität der Schaltung. Wenn es ein Hersteller aber wagen sollte, das IBM-BIOS-ROM unautorisiert zu kopieren, dann wird er sich alsbald mit IBM vor dem Richter wiedersehen. Trotzdem kann bei völlig anderer Realisation des BIOS funktionelle Gleichheit erreicht werden. Dazu muß eine genau definierte Schnittstelle geschaffen werden, die beide Bereiche scharf voneinander trennt. Die Schnittstelle darf lediglich Funktion, Ein- und Ausgabe eines Hard- oder Softwaremoduls bestimmen und der Realisation muß sie völlige Freiheit lassen. Die BIOS-Interrups des IBM-PCs bilden genau eine solche Schnittstelle.

Solche Schnittstellen werden von den meisten Software-Produzenten respektiert. Sie hoffen, daß ihre Programmen auch mit späteren, verbesserten Hardware-Konfigurationen laufen, wenn nämlich der Hersteller bei seinen Verbesserungen seine von ihm geschaffenen Schnittstellendefinitionen ebenfalls respektiert. Außerdem ist die saubere Trennung zwischen logischen und physikalischen Programmteilen eine der Grundanforderungen an gute Programmierung. Gefahr für eine strenge Trennung besteht immer dann, wenn eine Schnittstelle nicht umfassend genug gestaltet ist. Beim IBM-PC sind zum Beispiel die BIOS-Routinen zur Bedienung der seriellen Schnittstelle äußerst restriktiv in der Behandlung der Steuerleitungen beschaffen. Deshalb versuchen Software-Produzenten die Schwächen der Routinen zu umgehen um die Steuerleitungen besser in den Griff zu bekommen. Anstelle unschöner Quereinstiege in die BIOS-Routinen zu versuchen, haben sie Kommunikationsprogamme geschrieben, die direkt auf den Hardware-Baustein zugreifen, der die Steuerleitungen treibt. Das ist ein Baustein vom Typ 8250. Die Folge ist, daß es keiner der Hersteller kompatibler Geräte wagen kann, diesen nun doch schon angejahrten Baustein durch einen besseren Baustein zu ersetzen.

Wie schon gesagt, benutzt Digital Research die notwendigerweise verschiedenen Copy-Right-Einträge der verschiedenen Hersteller im BIOS, um die Computertypen zu unterscheiden. Dabei wird eine Schnittstelle verwendet, die nun überhaupt nicht definiert ist: das Auslesen des BIOS-ROMs.

Eine exakte Definition des Begriffes Kompatibilität sollte nicht mit fadenscheinigen Prozent-Angaben operieren, sondern die Schnittstellen genau angeben, für die funktionale Gleichheit herrscht. Dann wäre zu sagen, daß die IBM-Kompatiblen hinsichtlich der BIOS-Interrups kompatibel sind und in der Hardware identisch – soweit es die Software feststellen kann (ob Speicherselektierung mit PALs oder diskret ist für die Software unerheblich).

Analyse

Nach diesem Ausflug mit realem Hintergrund wieder zur Anpassung von GEM, die bei meiner Version von GEM mit der Veränderung der Copy-Right-Abfrage gelang. Wie wurde es gemacht? Ansatz der Analyse bildete der ausgegebene Text. War der erst einmal gefunden, und der Zeiger, der auf ihn zeigt, so ist die ganze Arbeit schon getan. Beides hätte Digital Research beliebig verschleiern können, aber sah dazu wohl keine Veranlassung.

Jeder kann mit Debug leicht verfolgen, was bei einer solchen Analyse zu tun ist. Und sollte eine neuere Version von GEM in den Handel kommen, kann das Verfahren sicher leicht wiederholt werden. Meine GEM-Version, auf die sich hier alles bezieht, hat die Versionsmeldung 1.22D, vom 13. 5. 86. Zunächst wird der Debugger (*Bild 1*) zusammen mit GEMV-

```
program gemzap; {zapt GEMVDI.EXE für IBM-PC-Kompatible}

label 1,2;

const quersumme_deutsch = - 25215;   {Quersumme deutsche Version}
      quersumme_englisch = - 22657;  {Quersumme englische Version}
      quersumme_englisch_12 = 27346;
      quersumme_deutsch_12 = 31552;
      quersumme_deutsch_12_neu = -27734;
      quersumme_deutsch_122 = -27844;
      zap_differenz = 184;           {Änderung der Quersumme nach Zappen}
      zap_differenz_12 = -12;        {Änderung der Quersumme V 1.2 u}
      zap_differenz_d12_neu = $77;   {Änderung der Quersumme V 1.2 deutsch neu
                                      = $EB (JMP) - $74 (JZ) }
      pos_10  = 6928;                { Position des Zaps englisch 1.0 }
      pos_12U = 5092;                { dito englisch 1.2}
      pos_12D = 5108;                { dito deutsch 1.2}
      pos_12D_neu = $1401;           { dito deutsch 1.2 neu (Jan 1986) }

type vtyp = (unbekannt,englisch,deutsch,englisch_12,deutsch_12,deutsch_12_neu,
             deutsch_122);

var gemvdi:file of byte;
    zeichen,jmp,jmp1,jmp2,jmp3,nop,xor1,xor2:byte;
    ch:char;
    i,summe,gueltige_quersumme,pos,laenge:integer;
    version:vtyp;

begin
   writeln ('*** GEMZAP.COM ***');
   writeln ('V1.6  13.5.86     ');
   writeln ('(c) 1985 Röckrath Mikrocomputer, 5100 Aachen');
   writeln;
   writeln ('Modifiziert GEM System Master Disk für IBM-Kompatible PC''s');
   writeln ('-----------------------------------------------------------');
   writeln ('( Modifiziert englische und deutsche Version 1.0/1.2 von GEM! )');
   writeln;
   writeln ('Wenn Sie einen IBM-PC oder IBM-PC/XT besitzen, brauchen Sie');
   writeln ('GEMZAP.COM nicht zu benutzen.');
   writeln;
   writeln ('Haben Sie einen IBM-PC oder IBM-PC/XT (ja = j, nein =n) ?');
   ch := ' ';
   while (ch<>'j') and (ch<>'n') and (ch<>'J') and (ch<>'N') do
      begin
         read (ch);
         if ch='j' then goto 1;
      end;
2: writeln;
   writeln ('Bitte legen Sie die >Kopie< der GEM System Master Disk in Laufwerk
A: ein');
   writeln ('und betätigen Sie die <RETURN>-Taste.');
   writeln;
   read (ch);
 assign (gemvdi,'gemvdi.exe');
 {$I-}
 reset (gemvdi);
 {$I+}
 if ioresult <> 0 then
    begin
       write ('>>> Sie haben nicht die richtige Diskette eingelegt ! <<<');
       goto 2
    end;
```

Bild 2. Dieses Turbo-Pascal-Programm macht's automatisch

```
    summe := 0;
    version := unbekannt;
    nop := $90;            {Hexcode für Maschinenbefehl NOP}
    xor1:= $30;xor2:=$e4;   {Hexcodes für XOR AH,AH}
    jmp := $eb;

    for i:=1 to filesize(gemvdi) do
       begin
          read (gemvdi,zeichen);
          summe := summe + zeichen;
       end;
    close (gemvdi);
    reset (gemvdi);

    if (summe = quersumme_englisch) or
       (summe = quersumme_englisch + zap_differenz)
          then version := englisch;
    if (summe = quersumme_deutsch) or
       (summe = quersumme_deutsch + zap_differenz)
          then version := deutsch;
    if (summe = quersumme_englisch_12) or
       (summe = quersumme_englisch_12 + zap_differenz_12)
          then version := englisch_12;
    if (summe = quersumme_deutsch_12) or
       (summe = quersumme_deutsch_12 + zap_differenz_12)
          then version := deutsch_12;
    if (summe = quersumme_deutsch_12_neu) or
       (summe = quersumme_deutsch_12_neu + zap_differenz_d12_neu)
          then version := deutsch_12_neu;
    if (summe = quersumme_deutsch_122) or
       (summe = quersumme_deutsch_122 + zap_differenz_d12_neu)
          then version := deutsch_122;

    if version in [deutsch_12_neu,deutsch_122]
       then laenge := 8192
       else laenge := 7680;

    if (version = unbekannt) or (filesize (gemvdi) <> laenge) then
       begin
          writeln ('››› Diese GEM System Master Disk kann nicht modifiziert
werden! ‹‹‹');
          goto 1
       end;

    case version of
       englisch,deutsch:              pos:=pos_10;
       englisch_12:                   pos:=pos_12u;
       deutsch_12:                    pos:=pos_12d;
       deutsch_12_neu,deutsch_122:    pos:=pos_12d_neu;
    end;
    seek (gemvdi,pos);
    read (gemvdi,jmp1,jmp2,jmp3);
    close (gemvdi);
    write ('In Laufwerk A: befindet sich die ');
    case version of
       englisch:      write ('englische Version 1.0');
       deutsch:       write ('deutsche Version 1.1');
       englisch_12:   write ('englische Version 1.2U');
       deutsch_12:    write ('deutsche Version 1.2D');
       deutsch_12_neu:write ('deutsche Version 1.2D (ab 1986)');
       deutsch_122   :write ('deutsche Version 1.22D');
    end;
    writeln (' von GEM !');
    writeln;
    case version of
       englisch:      gueltige_Quersumme :- quersumme_englisch;
       deutsch:       gueltige_Quersumme :- quersumme_deutsch;
       englisch_12:   gueltige_Quersumme :- quersumme_englisch_12;
       deutsch_12:    gueltige_Quersumme :- quersumme_deutsch_12;
       deutsch_12_neu:gueltige_Quersumme :- quersumme_deutsch_12_neu;
       deutsch_122   :gueltige_Quersumme :- quersumme_deutsch_122
    end;
    case version of
       englisch,deutsch:
          begin
             if (summe = gueltige_Quersumme + zap_differenz) and
                (jmp1 = nop) and (jmp2 = nop) and (jmp3 = nop) then
                begin
                  writeln ('››› Diese GEM System Master Disk ist bereits
modifiziert ! ‹‹‹');
                  goto 1
                end;
          end;
       englisch_12,deutsch_12:
          begin
             if (summe = gueltige_Quersumme + zap_differenz_12) and
                (jmp1 = xor1) and (jmp2 = xor2) then
                begin
                  writeln ('››› Diese GEM System Master Disk ist bereits
modifiziert ! ‹‹‹');
                  goto 1
                end;
          end;
       deutsch_12_neu,deutsch_122:
          begin
             if (summe = gueltige_Quersumme + zap_differenz_d12_neu) and
                (jmp1 = jmp) then
                begin
                  writeln ('››› Diese GEM System Master Disk ist bereits
modifiziert ! ‹‹‹');
                  goto 1
                end;
          end;
    end;
    if summe = gueltige_Quersumme then
       begin
          reset (gemvdi);
          seek (gemvdi,pos);
          case version of
             englisch,deutsch:
                begin
                   if (jmp1 = $E9) and (jmp2 = $0F) and (jmp3 = $00) then
                      begin
                         write (gemvdi,nop,nop,nop);
                      end;
                end;
             englisch_12,deutsch_12:
                begin
                   if (jmp1 = $90) and (jmp2 = $90) then
                      begin
                         write (gemvdi,xor1,xor2);
                      end;
                end;
             deutsch_12_neu,deutsch_122:
                begin
                   if (jmp1 = $74) then
                      begin
                         write (gemvdi,jmp);
                      end;
                end;
          end;
          close (gemvdi);
          writeln ('Ihre GEM System Master Disk ist nun zum Betrieb auf einem
IBM-Kompatiblen');
          writeln ('(wie z. B. Olivetti M24, Commodore PC-10, TCS-Genie 16 C)
modifiziert.');
          writeln;
          writeln ('Bitte starten Sie nun die Installation mit GEMPREP.');
       end
    else
       writeln ('››› Diese GEM System Master Disk kann nicht modifiziert werden!
‹‹‹');
1:
end. {gemzap}
```

DI.EXE von der GEM System Master Disk geladen. Dieses Programm, das Virtuell Device Interface, beinhaltet die Abfrage. Dann sollte DS den Wert von CS bekommen, damit sich der D- und U-Befehl des Debuggers auf das gleiche Segment beziehen. Der Such-Befehl mit dem Meldetext als Argument ergibt die Adresse des Textes, die ihrerseits nach Vertauschen ihres höherwertigen Bytes mit dem niederwertigen Byte zum Gegenstand der Suche wird. Und wie erwartet findet sich die Adresse in einem Ladebefehl für das Register DX. Der Kenner ahnt hier schon, daß ein MS-DOS-Aufruf Nr. 9 folgen muß, der den Text ausgeben soll. Diese Stelle ist also ganz heiß. Disassembliert man ein paar Zeilen vor dieser Sequenz, so sieht man sofort, wie man eingreifen kann: Durch Wortvergleichsbefehle werden die ersten zwei Zeichen des Herstellernamens abgefragt. Akzeptiert werden IB für IBM und CO für Compaq. Besitzer dieser Computer dürfen also ganz legal mit GEM arbeiten. Nach jeder dieser drei Abfragen folgt ein JZ, der bei erfolgreicher Copy-Right-Prüfung in den Rest des Programmes führt, bei einem Mißerfolg aber mit der Adresse des Meldetextes in DX an eine Stelle verzweigt (CS:10FD), die ihn dann ausgibt und danach zum DOS zurückkehrt. Da dieser Programmteil unter verschiedenen Abbruchbedingungen mit verschiedenen Texten angesprungen wird, befindet sich der verräterische Ladebefehl in unmittelbarer Nachbarschaft der drei Abfragen. Dies hat die Suche ungemein erleichtert.
Die Änderung ist jetzt genauso einfach. Einer der drei JZ-Befehle wird durch einen unbedingten JMP (=EBH) ersetzt. Wer aber meint, mit dem E-Befehl im Debugger die Änderung durchführen zu können , war etwas voreilig. Das Abspeichern ist bei .EXE-Programmen im Debugger nicht möglich. Aber ein anderes verbreitetes Hilfsprogramm ermöglicht die kleine Änderung im Handumdrehen, wenn man einen kleinen Kniff beachtet. Bei .EXE-Programm-Dateien entspricht die Adresse 1201H, auf der wir den neuen Sprung einfügen wollen, der relativen Byte-Adresse 1401H auf der Diskette. Also muß das zweite Byte im zehnten Sektor auf der Diskette geändert werden (ein Sektor hat 512 Bytes, die Sektorzählung beginnt bei 0). Jetzt kann man mit den Norton-Utilities oder einem anderen Diskettneditor die Änderung schnell probieren und GEM neu und erfolgreich starten.
Die Änderung kann übrigens auf einer Kopie der System Master Disk oder auf einer installierten Version stattfinden. Nach der Installation findet man eine Kopie des GEMVDI.EXE im Inhaltsverzeichnis GEMSYS auf der Startup-Diskette oder Festplatte.

Das ZAP-Programm in Turbo-Pascal

Das Programm in *Bild 2* erledigt, was im letzten Abschnitt noch von Hand geschah. Außerdem tut es dies für insgesamt 7 Versionen. Zeitweise schien es, als ob Digital Research wöchentlich neue GEM-Versionen herausbringen wollte. Wobei auch in den neuen Versionen immer wieder noch kleine Fehler zu finden waren, die zwar oft nicht erheblich aber ärgerlich waren. Beispielsweise gab ein bestimmte Version gern 13465732345 Bytes freien Speicher auf der Diskette an, was wirklich eindrucksvoll wäre und eine Revolution der Datenkomprimierung bedeuten würde. Andere Versionen verschoben Texte, die in Grafiken eingebaut waren, wenn alles zusammen auf dem Drucker ausgegeben wurde. Die Krönung waren zwei Versionen, von denen eine bei der Installation schon ausstieg und die andere beim Laden der installierten Software. In beiden Fällen hatte man keine Möglichkeit, GEM überhaupt anzuwerfen. Die letztgenannte Version lief aber immer dann, wenn die installierten Programme nicht von der Diskette sondern vom Festplattenlaufwerk kam. Man hielt es wohl nicht für notwendig, die Software auch von Disketten her vor der Auslieferung zu überprüfen. Allerdings sei aus Gerechtigkeitsgründen zum Schluß erwähnt, daß die neueste mir bekannte Version (Desktop V1.22D) von solchen Fehlern frei zu sein scheint.
Das Turbo-Pascal-Programm versucht hauptsächlich die verschiedenen Versionen zu identifizieren, denn jede erfordert eine andere Änderung. In einer ersten Version wurden zum Beispiel ganz trickreich die Codes der Hersteller-Buchstaben einer Exclusiv-Oder-Verknüpfung unterzogen und auf deren Ergebnis hin abgefragt. Die Version ermittelt das Programm anhand der Prüfsumme aller Programmbytes, die es zunächst errechnet. Dann prüft es, ob der GEMVDI schon modifiziert ist und modifiziert ihn gegebenenfalls.

Bilder mit dem Mac

Auf dem MacIntosh von Apple läuft ein professionelles Grafik-Gestaltungsprogramm, das bei der Arbeit an Präsentationen, an Druckvorlagen und an zeichnerischen Kunstwerken sehr hilfreich sein kann. Es heißt Adobe Illustrator, kommt aus den USA, kostet etwa 1500,– DM und wird von der Firma ALSO-ABC, Hamburg und München, importiert. Es arbeitet gern mit Pagemaker und Postscript zusammen und kann seine Ergebnisse an Laserdrucker und – das ist der Clou – auch an Linotypegeräte (Laserbelichter) abgeben, die für den Auflagendruck Filme mit großer Qualität erzeugen. Dabei können bis zu 2400 Pixel/Inch erreicht werden. Während die von Laserdruckern erzeugten Vorlagen für Handbücher, kleine Drucksachen und Vereinszeitschriften in der Qualität ausreichen, sind die von den Linotypegeräten hergestellten Filme für alle Arten von Druckerzeugnissen mit großer Auflage geeignet.

Adobe Illustrator wird mit Videodemonstration geliefert

Matthias Jordan

Mandelbrot-Menge schnell berechnen

Arithmetikprozessor 8087 macht dem PC Dampf

Mit dem Arithmetikprozessor 8087 läßt sich die Mandelbrot-Menge wesentlich schneller berechnen. Um die Leistungsfähigkeit des Arithmetikprozessors voll auszunutzen, hat unser Autor die rechenaufwendige Iterationsschleife in Assembler geschrieben und in ein Turbo-Pascal-Programm eingebunden.

Das Programm Fractals läuft auf einem PC, der mit einer Hercules-Karte und dem Arithmetikprozessor 8087 ausgestattet ist. Zusätzlich wird Turbo-Pascal mit der Implementierung der 8087-Befehle (Turbo 87) und die Turbo-Graphix-Toolbox benötigt. Normales Turbo-Pascal ist nicht einsetzbar, da dieses ein internes Fließkomma-Format verwendet, das nicht mit dem Arithmetikprozessor 8087 verträglich ist. Drei Include-Dateien am Programmanfang beziehen sich auf die Toolbox und stellen die Prozeduren für die Grafikbefehle zur Verfügung. Die Prozedur dp(Spalte, Zeile) setzt einen Punkt in der aktuellen Vordergrundfarbe an die angegebene Bildschirmkoordinaten-Position. Außerdem werden noch die Prozeduren hardcopy und loadscreen bzw. savescreen benutzt.
Das Ergebnis des Programms ist in *Bild 1* zu sehen.

Iterationsvorschrift

Man nehme eine komplexe Zahl $z_i = a + b \times j$ ($a = Re(z_i)$: Realteil, $b = Im(z_i)$: Imaginärteil, j = Imaginäre Einheit), quadriere sie, addiere eine Konstante dazu und erhält eine neue komplexe Zahl $z_{i+1} = z_i \times z_i + c$. Außerdem gibt man für z_o einen Startwert vor, der hier 0 gewählt wird. Die Konstante wird während der Iteration nicht verändert. Diese Iterationsvorschrift wendet man theoretisch unendlich oft an. Das Ergebnis ist eine Serie von komplexen Zahlen, die man eine Zahlenfolge oder kurz Folge nennt. Die Folge kann zwei grundsätzlich verschiedene Verhaltensweisen zeigen. Entweder stabilisieren sich die Folgenglieder mit wachsendem i auf einen festen Wert (dann ist die Folge konvergent) oder sie werden immer größer und wachsen über alle Grenzen (divergent). Das Programm Fractals (*Bild 2*) prüft, ob die Folge für ein bestimmtes c konvergiert. Dabei müssen einige Vereinfachungen getroffen werden. Es wird höchstens 300mal iteriert (anstatt unendlich). Tritt bis dahin kein Abbruch wegen drohendem Überlauf ein, wird die Folge als konvergent angesehen. Sobald ein Folgenglied betragsmäßig größer ist als 10 (im Programm wird das Quadrat betrachtet; der Wert 10 ist ein Erfahrungswert), ist mit einem Überlauf zu rechnen und die Iteration wird abgebrochen. Die Folge ist divergent. Stellt c einen Punkt in der komplexen Zahlenebene dar, so kann für jedes c das Konvergenzverhalten getestet und abhängig vom Ergebnis dieser Punkt gesetzt oder gelöscht werden. Das Programm Fractals geht noch weiter: es wertet den Zählerstand, der ein Maß für die Konvergenzgeschwindigkeit ist, beim Abbruch aus. War die Zahl ungerade, wird ein Punkt gesetzt, andernfalls nicht (das ist eine Modulo-2-Division). Jedem Zählerstand kann sogar ein bestimmter Farb- oder Grauwert zugeordnet werden (nicht bei Hercules-Grafik). Dazu führt man eine Modulo-n-Division durch, wobei n die Anzahl der verfügbaren Farben ist. Erst jetzt kann dem Ergebnis, das eine Zahl zwischen 0 und n − 1 ist, ein Farbwert zugeordnet werden. Hier bietet sich dem Programmierer ein weites Betätigungsfeld an, die eindrucksvollsten Farbkombinationen zu finden.
Die wichtigsten Variablen sind im Text erklärt. Das Programm ist auch mit anderen Grafikkarten lauffähig, sofern sie von der Toolbox unterstützt werden (Hercules, IBM-Farbgrafik, M24, Zenith-Farbgrafik).
cx ist der Realteil von c und cy der Imaginärteil; r ist das Quadrat des Radius, der die eine Abbruchbedingung festlegt. Tmp dient als Zwischenspeicher für das 8087 Statuswort und rr ist

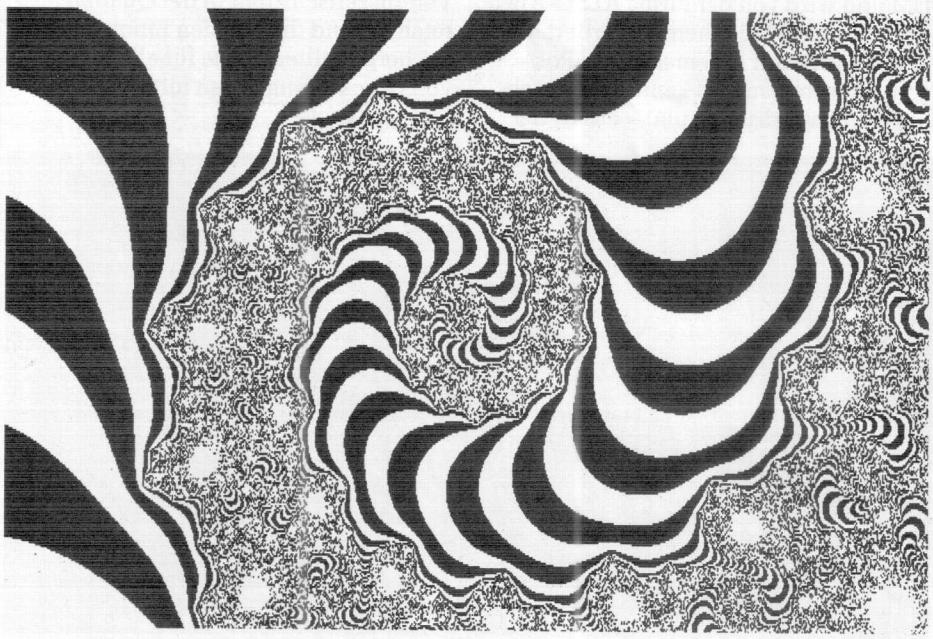


Bild 1. Das Seepferdchental der Mandelbrot-Menge

der Zählerstand beim Abbruch. Nach jeder gerechneten Zeile kann das Programm unterbrochen werden; das bis dahin gerechnete Bild wird unter FRACTALS.PIC im aktuellen Directory gespeichert. Außerdem wird die Nummer der zuletzt bearbeiteten Zeile auf dem Schirm ausgegeben. Nach einer Unterbrechung kann man wieder fortfahren, wenn man nach dem Programmstart als erste Zeile die 'Letzte+1' eingibt. Ist dieser Wert ungleich 1, wird das bisher gerechnete Bild geladen und am Schirm angezeigt.

Der Algorithmus wurde im Prinzip bereits in mc veröffentlicht [2], jedoch sinkt bei dieser Version bei erheblich vergrößerter Auflösung die Rechenzeit je nach Bereich im Apfelmännchen auf etwa 60 bis 90 Minuten bei 7,2 MHz Taktfrequenz (PC mit V20). Wird Turbo-87 ohne den Assemblerteil verwendet, steigt die Rechenzeit bei gleichen Abbruchbedingungen auf acht Stunden und mehr an!

Die Werte mittex und mittey sind auf das Zentrum eines Seepferdchentales eingestellt. Andere Bereiche findet man in [1], [2] und [3]. Zu beachten ist jedoch, daß der Wert laengex bei der Herculeskarte etwa das 1.5fache von laengey sein sollte, damit es keine Verzerrungen gibt. Der exakte Wert hängt von der individuellen Konfiguration ab und – bei Verwendung eines Druckers für die Hardcopy – vom Drucker.

Drei Include-Dateien

typedef.sys enthält die wichtigsten globalen Typen-, Variablen- und Konstantenvereinbarungen für alle Include-Dateien der Graphix-Toolbox.

graphix.sys stellt die hardwareabhängigen Grafik-Prozeduren zur Verfügung (Initialisierung der Grafik-Karte, Punkt setzen/löschen/testen ...). Im Lieferumfang der Graphix-Toolbox befinden sich u. a. graphix.hgc (Hercules), graphix.oli (M24) und graphix.ibm (IBM-Standardgrafik).

kernel.sys enthält die hardwareunabhängigen Teile der Graphix-Toolbox (Linien/Ellipsen/Kreise zeichnen ...) und baut auf graphix.sys auf.

Literatur

[1] *Klausberg, Dr. Karl:* PC-Magazin 1986, Heft 41, Seite 31.

[2] *Lautenbacher, Markus:* Apfelmännchen in Pascal. mc 1986, Heft 12, Seite 76.

[3] *Klein, Rolf-Dieter:* Floating Point Unit 68881. mc 1986, Heft 8, Seite 92.


```
program fractals;
{$i typedef.sys}
{$i graphix.sys}
{$i kernel.sys}
label 9999;
{  Konstanten aus Graphix Toolbox:                                          }
{    XScreenMaxGlb    Anzahl der Punkte pro Zeile-1  (hercules: 719)        }
{    YMAXGLB          Anzahl der Zeilen-1            (hercules: 347 ... 351) }

var r,cx,cy,dx,dy,mittex,mittey,laengex,laengey:real;
   zeile,spalte,zeile1,rr,tmp:integer;

begin;

mittex:=-0.7451569;
laengex:=1.5e-4;
mittey:=0.112575574;
laengey:=1e-4;

r:=100;                                          {Radius für Abbruchbedingung}
zeile1:=1;                              {Default für erste zu bearbeitende Zeile}

clrscr;
write('Erste Zeile:  ',zeile1,'  ');
readln(zeile1);                          {nächste zu bearbeitende Zeile eingeben}

initgraphic;                                    {Graphix-Toolbox initialisieren}

if zeile1<>1 then loadscreen('fractals.pic');    {ggf. angefangenes Bild laden}

drawborder;                                                  {Rahmen zeichnen}
dx:=laengex/XScreenMaxGlb;                  {Pixelabstand horizontal berechnen}
dy:=laengey/YMaxGlb;                         {Pixelabstand vertikal berechnen}

for zeile:=zeile1 to YMaxGlb-1 do                             {Zeilenschleife}
   begin;
   cy:=mittey+laengey/2-zeile*dy;

   for spalte:=1 to XScreenMaxGlb-1 do                       {Spaltenschleife}
      begin;
      cx:=mittex-laengex/2+spalte*dx;
      inline (
                     {init:                              Initialisierung        }
      $b9/$2c/$01/   { mov cx,012c  012ch=300d           Max. Anz. d. Iterat. ld. }
      $9b/$db/$e3/   {     finit                         8087 initialisieren    }
      $9b/$dd/$06/cx/{     fld      qword ptr [cx]       Realteil von C laden   }
      $9b/$dd/$06/cy/{     fld      qword ptr [cy]       Imaginärteil von C laden }
      $9b/$dd/$06/r/ {     fld      qword ptr [r]        Radius für Abbruchb. ld. }
      $9b/$d9/$ee/   {     fldz                          Null laden             }
      $9b/$d9/$f6/   {     fdecstp                       Stackptr. dekrementieren }
      $9b/$d9/$f6/   {     fdecstp                       Stackptr. dekrementieren }
      $9b/$d9/$ee/   {     fldz                          Null laden             }
      $9b/$d9/$ee/   {     fldz                          Null laden             }
                     {iterate:                           Iterationsschleife     }
      $9b/$d8/$cc/   {     fmul     st,st(4)             Y <- Y*X     gemischtes }
      $9b/$d8/$c0/   {     fadd     st,st(0)             Y <- Y+Y      Produkt   }
      $9b/$d8/$c6/   {     fadd     st,st(6)             Y <- Y+CY    Y ist fertig}
      $9b/$dd/$d2/   {     fst      st(2)                Y2 <- Y      kopieren  }
      $9b/$dc/$ca/   {     fmul     st(2),st             Y2 <- Y2*Y   quadrieren }
      $9b/$d9/$f7/   {     fincstp                       Stackptr. inkrementieren }
      $9b/$d8/$c6/   {     fadd     st,st(6)             Z <- Z+X     X ist fertig}
      $9b/$dd/$d3/   {     fst      st(3)                X <- Z       kopieren  }
      $9b/$d8/$c8/   {     fmul     st,st(0)             Z <- Z*Z     = X*X     }
      $9b/$dd/$d2/   {     fst      st(2)                X2 <- Z       kopieren }
      $9b/$d8/$e1/   {     fsub     st,st(1)             Z <- Z-Y2    Z ist fertig}
      $9b/$d9/$f7/   {     fincstp                       Stackptr. inkrementieren }
      $9b/$d8/$c1/   {     fadd     st,st(1)             Y2 <- Y2+X2  Radius    }
      $9b/$d8/$d3/   {     fcom     st(3)                Y2 mit R vergleichen   }
      $9b/$dd/$3e/tmp/{    fstsw    [tmp]                akt. Statuswort speichern}
      $9b/$d9/$f6/   {     fdecstp                       Stackptr. dekrementieren }
      $9b/$d9/$f6/   {     fdecstp                       Stackptr. dekrementieren }
      $9b/$8a/$26/
           tmp+1/    {mov  ah,[tmp+1]                   Status nach 8086/8088 -ah}
      $9e/           {sahf                              Status nach Flagregister }
      $77/$02/       {ja   exit                         Y2>R ===> Abbruch      }
      $e2/$c1/       {loop iterate                      CX<>0 ===> iterate     }
                     {exit:                                                    }
      $89/$0e/rr);   {mov  [rr],cx                      Zählerstand speichern  }

      if (rr and 1)<>0 then dp(spalte,zeile);                 {ggf. Pixel setzen}
      end;
   if keypressed then goto 9999;          {Abbruch durch Bediener ermöglichen}
   end;
9999:;
repeat until keypressed;                                  {Bild noch stehen lassen}
savescreen('fractals.pic');              {Bild auf aktuelles directory speichern}
leavegraphic;                                          {Auf Textmodus umschalten}
clrscr;
writeln('Letzte Zeile:  ',zeile);        {Nummer der zuletzt bearbeiteten Zeile}
end.
```

Bild 2. Turbo-87 und der Arithmetikprozessor 8087 sorgen für eine hohe Rechengeschwindigkeit

Dieter Strauß

Producer

Datenbanken ohne Programmierkenntnisse erstellen

Zu den zahlreichen Abkürzungen, die mit C anfangen (CAD, CIM usw.) ist eine neue hinzugekommen: CASE (Computer Aided Software Engineering). Producer ist ein CASE-Programm, mit dem sich jeder Anwender individuelle Datenbanken schaffen kann.

Producer von CAT-COMP, Saarbrücken, ist ein Programmgenerator für Datenbanken. Das Programm fragt den Anwender nach seinen Wünschen und erhält so alle Informationen, um selbständig Programme zu erstellen. Alle notwendigen Eingabe-, Such- und Druckroutinen sind in dem erzeugten Datenbankprogramm enthalten.

Wie wird Producer bedient?

Das Programm ist vollständig menügeführt. Der Anwender muß nur wissen, was in der Datenbank stehen soll. Nach dem Programmstart erscheint die Einschaltmeldung und die Frage, wie das Programm heißen soll. Vom System wird der eingegebene Programmname mit der Meldung „Neues Datenbank-System!!!" quittiert. Auf dem Bildschirm erscheint daraufhin die Maske zur Spezifizierung des gewünschten Programms.

Die Masken können in Kartei- und Tabellenform angelegt werden. Um eine Kartei anzulegen, wird der Cursor auf die gewünschte Bildschirmposition gesetzt und die Tastenkombination Alt-K eingegeben. An der Cursorposition kann jetzt die gewünschte Feldbezeichnung, die maximal 15 Zeichen enthalten darf, eingetippt werden. Mit der Tastenkombination Alt-I gelangt man in das Inhaltsfeld. Drei Platzhalter stehen für die Kartei- und die Tabellenform zur Verfügung. Z steht für Zeichenfeld, N für numerisches Feld und D für Datumsfeld (*Bild 1*).

Zur Gestaltung der Bildschirmseite können die Umrandungssymbole des IBM-Zeichensatzes verwendet werden. Die Tastenkombination Alt-G aktiviert diesen Modus. Unter diesem Menüpunkt lassen sich Überschriften und Zusätze wie DM, Stück oder % eingeben.

```
Kartei  Feld (Z/N./D)  unterstrichen       Gestalt       ESC = Befehlszeile
 Zeitschrift: ZZZZZZZZZZZZZZZZZZZZZZ     Jahrgang :  NNNN

 Heft-Nr    : NNN                        Seite    :  NNNN

 Titel, Beitrag: ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

 Nachträge:  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

 Stichwort:  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

 Autor : ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
 PLZ   : NNNNNN  Ort: ZZZZZZZZZZZZZZZZ  Straße  : ZZZZZZZZZZZZZZZZZZZZZZZZZZZ
 Tel   : ZZZZZZZZZZZZZZZZZZ             Postfach: NNNNNNNNNNNNNNNNN

 Firma : ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
 PLZ   : NNNNNN  Ort: ZZZZZZZZZZZZZZZZ  Straße  : ZZZZZZZZZZZZZZZZZZZZZZZZZZZ
 Tel   : ZZZZZZZZZZZZZZZZZZ             Postfach: NNNNNNNNNNNNNNNNN
 Telex : ZZZZZZZZZZZZZZZZZZ

 ALT: Kartei  Tabelle  Inhalt  Feld  Gestaltung  Move  Löschen  Attribut
```

Bild 1. Die Bildschirmmaske mit Kartei-Feldern und den Platzhaltern

Falls die Bildschirmmaske nicht der Vorstellung des Anwenders entspricht, so ist dies kein Beinbruch. Der Cursor wird einfach auf das entsprechende Feld gesetzt und Alt-M eingegeben. Mit dem Cursor läßt sich nun das Feld auf die gewünschte Bildschirmposition schieben. Beendet wird die Verschiebeoperation mit der Return-Taste.

Verbindungen zwischen einzelnen Bildschirmmasken lassen sich mit der Tastenkombination Alt-F herstellen.

Sind alle Masken fertig erstellt, so kann im Programm fortgefahren werden. Producer erstellt automatisch eine EXE-Datei.

Aufruf	Inhalt
+	Addition
-	Subtrakion
*	Multiplikation
/	Division
%	Divisionsrest
** oder ^	Potenzierung (hoch)
()	Klammern

Vergleichsoperationen

Aufruf	Inhalt
=	gleich
<	kleiner
>	größer
#	ungleich

Verknüpfungen

Aufruf	Inhalt
.NOT.	nicht
.AND.	und
.OR.	oder
XOR()	und/oder

Aufruf	Inhalt
PI	3.141592654
SI(X)	Sinus
SIH(X	Sinus-Hyperb.
ASI(X	Arcussinus
ITT(a,b)	Itteration
SU(a,b)	Summe von 2 Werten
ASUM(a,b,c)	Aufsummation
SQRT(X)	Quadratwurzel
LOG(X)	Logarithmus
GRAD(X)	RAD in GRAD
CO(X)	Cosinus
COH(X)	Cosinus-Hyperb.
ACO(X)	Arcuscosinus
ABS(X)	Absolutwert
EXP(X)	Exponentialfunkt.
RAD(X)	GRAD in RAD
TA(X)	Tangens
TAH(X)	Tangens-Hyperb.
ATA(X)	Arcustangens

Bild 2. Zahlreiche Funktionen sind anwendbar

Producer intern

Ein Blick hinter die Kulissen von Producer zeigt, daß das ganze kein Hexenwerk ist, sondern einfach bereits bestehende Elemente sinnvoll kombiniert und mit einem klugen Steuerprogramm ergänzt wurden.

Producer ist in dBaseIII geschrieben. Viele Funktionen wurden in Assembler und C neu entwickelt und in eine gesonderte Programmbibliothek eingebunden.

Zur Complierung wird der Clipper-Compiler und zum Erzeugen der EXE-Datei PLINK86 verwendet. Der Clipper-Compiler (Hersteller: Nantucet) und PLINK86 sind im Producer-Programmpaket nicht enthalten.

Producer erstellt aus interaktiv eingegebenen Masken anwenderspezifische Programme. Bis zu 27 Masken können in einem Programm erstellt werden. Pro Maske werden eine Datenbank und bis zu sieben Index-Dateien erstellt. Alle Datenbanken können zueinander in Relation gesetzt werden. Pro Record sind bis zu 99 Felder erlaubt. Die Größe der Datenbanken ist ausschließlich von der Größe der Festplatte abhängig. Zahlreiche Funktionen sind verwendbar (*Bild 2*).

Systemvoraussetzungen

Producer läuft auf PCs, die mit mindestens 512 KByte Hauptspeicher, einem Diskettenlaufwerk, einer Festplatte, MS-DOS 3.X und einer parallelen Schnittstelle ausgestattet sind.

Für wen eignet sich Producer?

Producer kann von jedem Anwender bedient werden. Da das System nicht gerade billig ist, rentiert sich Producer bei professionellen Programmierern, die individuelle Software schnell erstellen müssen. Die fertigen Programme liegen in einem für den Clipper-Compiler geeigneten Code vor und können damit in größere Programme mit eingebunden werden.

Die mit Producer erstellten Programme sind geistiges Eigentum des Erstellers.

14 MByte RAM auf einer Steckkarte — 80386 für 8088-PCs

Zwei besonders interessante Produkte hat Interquadram vorgestellt. Dabei handelt es sich zum einen um eine 14-MByte-Steckkarte für 80286/80386-PCs und zum anderen um ein 80386-Board, das in jedem 8088-PC eingesetzt für erheblich mehr Leistung sorgt.

MightyMeg 286/386 (*Bild 1*) heißt das Speicherboard, das für IBM-AT, -PC 286 XT sowie Compaq 386 gedacht ist.

Es erweitert die Speicherkapazität in Schritten von 265 KByte auf 3,5 MByte oder in Schritten von 1 MByte auf maximal 14 MByte. Die Platine ist mit Memory-Modulen in SIMM-Technologie (single inline memory module) bestückt, die nur etwa die Hälfte des Platzes bisheriger Speicherbausteine benötigen. Die 3,5-MByte-Version setzt SIM-Module ein, die zu 256 KByte x 9 organisiert sind. Bei Verwendung der 1-MByte-SIMM-Technologie wird ein maximaler Speicher von 14 MByte erreicht.

Der zusätzliche Speicher wird im split memory mapping eingesetzt. Damit wird zunächst der Systemspeicher auf 640 KByte aufgefüllt. Der verbleibende Speicher wird als RAM-Floppy oder Spooler genutzt. Die Installation der Platine ist sehr einfach, da Dip-Schalter oder Jumper nicht eingestellt werden müßen. Vielmehr wird die Karte automatisch konfiguriert. Eingesetzt werden kann diese Karte mit den Betriebssystemen Xenix, Unix, Topview und dem erweiterten DOS der 80286-PCs.

Auf einem mit 16 MHz getakteten Intel-Prozessor 80386 basiert die Karte Quad 386 XT (*Bild 2*), die auf einen eigenen Speicher bis zu 3 MByte zugreifen kann. Der 32-Bit-Speicher ist aus 256-KByte-DRAMs aufgebaut. Dazu kommen noch ein Video-Speicher mit 96 KByte und ein Cache-Speicher mit 32 KByte. Wie die Speichererweiterung konfiguriert sich auch diese Steckkarte automatisch.

Auf der Quad-386-XT-Karteist neben einem Sockel für den Coprozessor 80287 ein asynchrones Bus-Interface installiert. Die Expanded Memory Spezifikationen nach Lotus-Intel-Microsoft und Ast-Quadram-Ashton Tate werden voll unterstützt. Mit dieser Erweiterung können auf einem PC mehrere Programme parallel laufen.

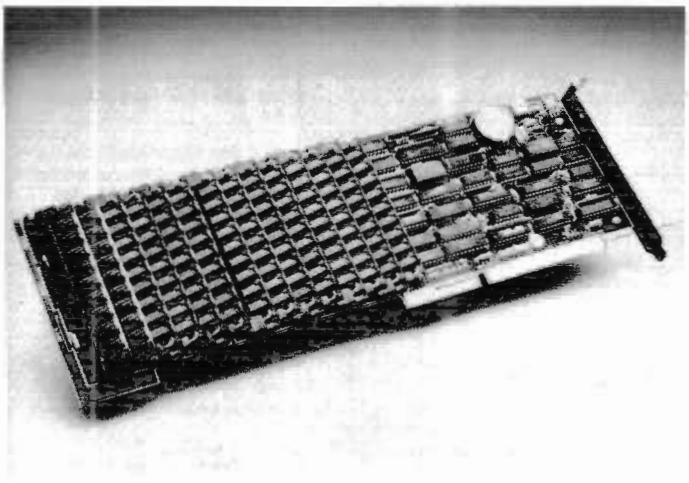


Bild 1. Mit bis zu 14 MByte kann man jetzt seinen 80286- oder 80386-PC erweitern

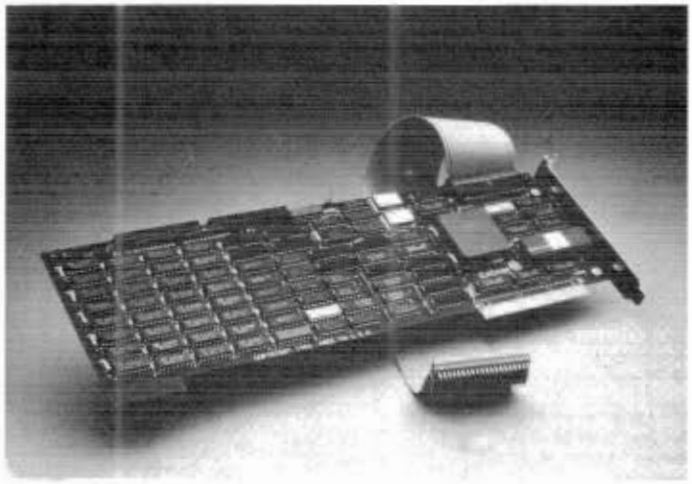


Bild 2. Für mehr Leistung im PC sorgt die 80386-Steckkarte von Interquadram

Reiner Schönrock

OS-9 für den Atari ST

Preiswertes Multitasking-/Multiuser-System

Lange hat es gedauert, bis aus der Ankündigung des Betriebssystems OS-9 für den Atari ST ein reales Produkt wurde. Rechtzeitig zur CeBIT konnte Atari die ersten Exemplare der von Microware angepaßten Software mit englischem Handbuch vorlegen. Wir haben uns OS-9 näher angesehen und dabei einige Erfahrungen gemacht. Für Profis und Schulen geeignet.

Für etwa 700 DM bekommt man neben dem Handbuch eine GEM-Boot- und eine OS-9-Systemdiskette und ein paar zusammengeklammerte Seiten, die über die Inbetriebnahme von Personal-OS-9 auf dem Atari ST informieren.
An Hardware setzt das Multitasking-Multiuser-Betriebssystem neben einem Atari ST nur ein Floppylaufwerk voraus. Auch mit einseitigen Floppylaufwerken ist der Betrieb von OS-9 denkbar. In diesem Fall bekommt man zwei einseitige Systemdisketten. Da OS-9 nicht nur alle Programme, sondern auch alle Betriebssystemkommandos, die es während der Arbeit benötigt, vom externen Speichermedium in den internen Speicher lädt und von dort ausführt, ist der Einsatz einer Festplatte empfehlenswert. Bei jedem Kommando wie „dir" (Inhaltsverzeichnis) oder „procs" (Übersicht der aktiven Prozesse) liest OS-9 vom externen Speicher das Programm ein, sofern es nicht konstant in den Systemspeicher geladen wurde. Und das immer wieder erforderliche Laden eines Moduls dauert bei den Floppylaufwerken des Atari ST zu lange für den professionellen Einsatz. Natürlich könnte man die wichtigsten Module gleich beim Booten in den Speicher laden. Das würde aber zu Lasten des freien Arbeitsspeichers gehen und den Startvorgang erheblich verlängern. Die Geschwindigkeit des Datentransfers zwischen Computer und Massenspeicher kann durch Puffer-Speicher erhöht werden. OS-9 ist hier sehr flexibel und passt sich den Anforderungen der Anwender an.
Wir gehen im folgenden davon aus, daß an den Atari ST eine Festplatte angeschlossen ist. Sie bietet einen guten Kompromiß zwischen Schnelligkeit und Belastung des Arbeitsspeichers. Auf einem Atari ST mit 1 MByte RAM ist es sinnvoll, eine RAM-Disk mit 256 KByte zu installieren und den Puffer-Speicher je Device-Deskriptor mit 64 KByte festzulegen. Dann bleibt immer noch genügend Speicher für die Arbeit mit mehreren Prozessen frei.

Vor dem Probieren installieren

Bevor man sich an die Installation des leistungsfähigen Betriebssystems macht, um z. B. die Multitasking-Fähigkeiten zu testen, sollte man sich zahlreiche Disketten bereitlegen und Sicherheitskopien aller Dateien der Festplatte machen, die man vielleicht irgendwann unter GEM/TOS noch einmal braucht.
Zu Beginn der Installation ist nämlich eine neue Aufteilung der Festplatte erforderlich, da OS-9 eine eigene Partition benötigt, die nicht von GEM aus angesprochen werden kann.
Die OS-9 Partition muß mit der GEM-Software, die mit der Festplatte geliefert wird, so vorbereitet werden, daß einige MBytes auf der Festplatte frei bleiben. Eine Aufteilung, die sich bei unserer 20-MByte-Festplatte SH204 bewährt hat ist:
C = 4,5 MByte
D = 5,2 MByte
E = 5,2 MByte.
Den restlichen Speicher von etwa 6,2 MByte haben wir frei (Unused) gelassen. In diesem Bereich wird OS-9 dann bei der Installation eingerichtet.
OS-9 würde sich auch mit weniger Platz zufriedengeben; laut einem Hinweis in FDISK sollen 100 KByte genügen.
Auf der mit dem OS-9-System gelieferten GEM-Boot-Diskette findet sich neben zwei Startprogrammen – eins für ein Festplattensystem und eins für ein Floppysystem – das Partitionierprogramm FDISK. Damit werden die von GEM nicht belegten Sektoren der Festplatte für OS-9 vorbereitet.
Aus dem ersten physikalischen Sektor der Festplatte liest FDISK die Anzahl der Köpfe (bei der SH204 sind es 4) und die Anzahl der Sektoren pro Track (17). Diese Angaben sollte man übernehmen; man kann sie auch ändern. Danach steht die Formatierung der Festplatte an, die bei der von uns gewählten Größe von 6,2 MByte einige Minuten dauert.
Zur Formatierung der Festplatte muß OS-9 gebootet werden. Dazu startet man das Programm „Startfd" der GEM-Boot-Diskette. Diese Diskette muß unbedingt in Laufwerk „A:" liegen, denn das 650 Bytes kleine Programm lädt die Systemdatei OS9FD.SYS nur von Laufwerk A.
Nach dem Start des Boot-Programms fordert das System die OS-9 Systemdiskette an, mit der dann OS-9 installiert wird. Auf dem Bildschirm wird angezeigt, welche Befehle während des Bootens ausgeführt werden und – dank ausführlicher Kommentare – welche Auswirkungen sie haben.
Wurde der Boot-Vorgang erfolgreich abgeschlossen, gibt man den Befehl „format /h0" ein. Als zu formatierendes Medium ist „/h0" für die Festplatte angegeben. Wie schon bei der Partitionierung zeigt das System zuerst die technischen Daten der Festplatte an. Dabei kommen jetzt aber doppelt soviele Sektoren pro Track zur Anzeige, da „format" einen Sektor mit 256 Bytes annimmt – im Gegensatz zu FDISK, das einen Sektor mit 512 Bytes erkennt. Am Ende dieses Vorgangs gibt das Programm die Anzahl der formatierten Sektoren aus (in unserem Beispiel 24480) und schreibt die Root-Directory-Struktur auf die Platte.
Auf der OS-9 Systemdiskette sind zwei Programme vorhanden, die die Erstellung von Systemdisketten und das Kopieren der Systemdiskette auf die Festplatte erleichtern. Durch das OS-9 Programm „install_hard_disk" wird ein Vorgang eingeleitet, der wieder mehrere Minuten dauert und – neben einer Kopie der Systemdateien – OS-9 von der Festplatte aus startfähig macht. Mit der Befehlsfolge „chd /h0; chx /h0/cmds" wird als aktuelles Laufwerk die Festplatte angegeben und das Directory, das die Systemkommandos enthält, mit CMDS angemeldet.
Bei jedem Aufruf eines Kommandos sucht OS-9 jetzt zuerst im Hauptspeicher nach dem ausführbaren Programm.

Wird es dort nicht gefunden, wird im Directory CMDS der Festplatte /h0 weitergesucht.
Über die dabei im Einzelnen ablaufenden Vorgänge haben wir bereits in unserer Serie über OS-9 in den Ausgaben 10/11/12/86 berichtet. Hier wollen wir nicht näher auf die internen Abläufe, sondern vielmehr auf einige Erfahrungen eingehen, die wir mit OS-9 auf dem Atari ST gemacht haben.

Aller Anfang ist schwer

Wer auf dem Atari ST ernsthaft mit OS-9 arbeiten möchte, wird nach dem Einschalten des Computers ohne den Umweg über das GEM sofort ins OS-9 gelangen wollen. Dazu ist das Programm „Starthd" in den Auto-Ordner einer GEM-Diskette zu kopieren. Auf das Root-Directory muß man die Datei OS9HD.SYS kopieren, die vom Start-Programm benötigt wird. Weitere Programme in den Auto-Ordner zu kopieren oder sogar Accessories laden zu wollen ist sinnlos – OS-9 kann nichts damit anfangen. Nach einem Reset oder dem Einschalten führt der Computer „Starthd" aus und bootet mit der Datei OS9HD.SYS das System von der Festplatte. Das Laden eines Festplattentreibers, wie es unter GEM erforderlich ist, entfällt.
Sollte das nicht auf Anhieb gelingen, ist mit der Befehlsfolge
„OS9GEN -Z=bootlist_for_H0"
das korrekte OS-9-Bootfile auf der Festplatte anzulegen. Die Bootliste ist in dem Verzeichnis der Kommandos (CMDS) im Unterverzeichnis „Bootobjs" zu finden. Hier sind auch alle Device-Deskriptoren gespeichert und können gegebenenfalls in die Bootliste eingetragen werden.

Ähnlich einer Autoexec-Datei unter MS-DOS kann man unter OS-9 eine Startup-Datei kreieren, die Funktionen automatisch ablaufen läßt. Besonders wichtig ist, daß man das aktuelle Datum richtig eingibt. OS-9 bietet hierzu das Kommando „setime", das nach dem Datum und der Uhrzeit fragt. Vergißt man die Zeitangabe, gibt ein falsches Datum ein oder bricht die Abfrage durch Ctrl-C ab, findet man sich bald in seinem System nicht mehr zurecht. Ohne eine Systemzeit zu definieren bekommt man z. B. bei der Auflistung aller im System ablaufender Prozesse nur solche mit einer „Lebenszeit" von 0.00 Sekunden. Ferner bekommen alle Dateien, die verändert werden, das „aktuelle" Datum 00.00.00.
Mit der Startup-Datei sollte man auch die Kommandos, die man sehr oft benutzt, in den Speicher laden. Dazu zählen DIR, PROCS, EDT (Zeileneditor) und ähnliche Utilities. Bei einem Atari ST mit 1 MByte RAM fallen diese kleinen Programme kaum ins Gewicht, bringen aber Geschwindigkeitsvorteile bei der Arbeit. Sollte am Atari ST nur ein Floppylaufwerk angeschlossen sein, kommt man um dieses Verfahren nicht umhin, da ja nicht immer die Diskette mit den Systemkommandos im Laufwerk liegen wird.

Von GEM nach OS-9

Möchte man von GEM aus direkt das OS-9 aufrufen, ohne eine Diskette zu wechseln, muß man die Datei OS9HD .SYS und das Programm „Starthd" zusammen auf eine GEM-Partition der Festplatte kopieren. Es empfiehlt sich, einen eigenen Ordner für OS-9 anzulegen. Das Startprogramm muß man an der Stelle ändern, an der auf „A:OS9HD-.SYS" zugegriffen wird. Dazu haben wir den Minimonitor genommen, den wir in der letzten Ausgabe vorstellten. Nach dem Einlesen der Programmdatei in einen freien Speicherbereich haben wir nach der entsprechenden Zeichenfolge gesucht und aus „A:" ein „C:" gemacht. Mit dieser kleinen Änderung war es nun möglich, von der Festplatte unter GEM aus das OS-9 ohne Systemdisketten hochzufahren. Hat man unter GEM bereits Datum und Zeit eingegeben, kann man in OS-9 durch „setime -s" die Systemzeit des Atari ST für OS-9 übernehmen.

Die wichtigsten Kommandos

Bei den ersten „Gehversuchen" mit OS-9 kommt man mit dem Kommando „dir" in Berührung. Es gibt einen Überblick über die Dateien, die auf dem aktuellen Laufwerk vorhanden sind.
Unterverzeichnisse kann man unter OS-9 daran erkennen, das sie vereinbarungsgemäß mit Großbuchstaben geschrieben werden. Dem Dir-Befehl kann man noch zahlreiche Optionen nachstellen (*Bild 1*). Durch die Angabe von „-?" werden zu jedem Befehl alle Optionen aufgelistet und Hinweise zum richtigen Gebrauch des Befehls gegeben. Gibt man „dir -e" ein, erhält man das in *Bild 2* wiedergegebene „Extended"-Inhaltsverzeichnis. Zusätzlich zur Größe der einzelnen Dateien in Bytes wird hier eine Liste der Attribute aufgeführt. Mit der Option „-r" erhält man ein Inhaltsverzeichnis, in dem alle Subdirectories mit einem Backslash gekennzeichnet werden. Die Attribute sind mit dem Kommando „attr" beliebig einstellbar.
Die Hardcopy-Routine des Atari ST funktioniert auch unter OS-9. Möchte man die Ausgaben einiger Befehle auf den Drucker umleiten, ist der Zusatz „>/P" für das Device „Printer" erforderlich. Würde man stattdessen „>P" eingeben, würde OS-9 eine Datei „P" erzeugen und die Ausgaben auf das aktuelle Laufwerk schreiben.
Außer über die Dateien auf der Diskette oder Festplatte kann man sich noch über die im Systemspeicher befindlichen ausführbaren Module informieren. Dazu dient das Kommando „mdir" (*Bild 3*).

Multitasking

Um sich einen ersten Eindruck davon zu verschaffen, was Multitasking unter OS-9 heißt, kann man dem Kommando „mdir -e" den Zusatz „>/P" für Umleitung auf den Drucker und zusätzlich noch das Zeichen „&" anhängen. Dadurch erhält man auf dem Drucker eine

```
$ dir -?
Syntax: dir [<opts>] {<dir names> [<opts>]}
Function: display directory contents
Options:
     -a         show all files
     -d         show directories with a slash
     -e         extended dir listing
     -n         treat dirs like files
     -r         recursive dir listings
     -r=<num>   recursive dir listing to depth <num>
     -s         unsorted dir listing
     -u         unformatted listing
     -x         directory is execution dir
     -z         read dir names from stdin

$ dir -a
                    Directory of . 15:57:11
.login          CMDS            DEFS            LIB
OS9Boot         SYS             first           grep
konverter       rechne          startup         startup.back
```

Bild 1. OS-9 gibt zu vielen Kommandos Benutzungshinweise

```
$ dir -e
                    Directory of . 16:02:02
 Owner    Last modified Attributes Sector Bytecount Name
-------   ------------- ---------- ------ --------- ----
  0.0     00/00/00 2400  d-ewrewr     9E      1984 CMDS
  0.0     87/04/13 1525  d-ewrewr    806        64 DEFS
  0.0     87/04/13 1525  d-ewrewr    805        64 LIB
  0.0     00/00/00 2400  d-ewrewr    71B       160 MC68000
  0.0     87/03/20 1253  ------wr     16     34456 OS9Boot
  0.0     87/04/13 1525  d-ewrewr    71E       384 SYS
  0.0     87/03/23 1557  ------wr    71C       410 first
  0.0     87/03/26 1538  ------wr    803      5654 grep
  0.0     85/08/12 1631  ------wr    B23       912 init.ram
  0.0     87/03/24 1725  ------wr    719       542 konverte
  0.0     87/03/25 1321  ------wr    808       420 rechne
  0.0     00/00/00 2400  ----r-wr    718       290 startup
  0.0     87/03/20 1633  ----r-wr    773      1104 startup.
  0.0     00/00/00 2400  ------wr    749        59 test
```

Bild 2. So sieht das erweiterte Inhaltsverzeichnis einer Festplatte aus. Bei Dateien mit dem Attribut „d" handelt es sich um Subdirectories.

umfangreiche Liste aller im Systemspeicher liegender ausführbarer Module mit Startadresse, Umfang in Bytes, Typbezeichnung und Anzahl der Links (*Bild 4*), während man gleichzeitig andere Befehle auf dem System ausführen kann. Die Ausgabe des Inhaltsverzeichnisses auf den Drucker wird als neuer Task in das Prozedurverzeichnis aufgenommen (*Bild 5*). Der Prozeß mit der Identifikationsnummer 5 ist zum Zeitpunkt der Prozeßabfrage aktiv und hat bis dahin 0,14 Sekunden der CPU-Zeit beansprucht. Zusätzlich könnte man dem System nun noch den Auftrag geben, eine Datei über die serielle Schnittstelle (/t1) zu übertragen, z. B.:

list startup >/t1 &

Prozesse kann man mit dem Kommando „kill" wieder aus dem Verzeichnis der aktiven Prozesse entfernen, z. B. wenn der Empfänger an der seriellen Schnittstelle nicht antwortet.

Multiuser

Neben der gleichzeitigen Bearbeitung mehrerer Aufträge kann ein OS-9-System auch von mehreren Anwendern gleichzeitig benutzt werden.
Die Multiuser-Fähigkeiten haben wir mit einem Olivetti M24 getestet, der über ein Nullmodem (Dreidrahtleitung) mit dem Atari ST verbunden wurde und auf dem das Terminal-Programm PC-Term (Franzis Software Service) lief. Von dieser Seite aus sind nur die Übertragungsparameter wie Baudrate und Anzahl der Datenbits mit denen des Atari ST übereinstimmend einzustellen. Auf dem Atari ST unter OS-9 ist der Timesharing-Monitor „tsmon" zu aktivieren: „tsmon /t1 &".
Vom PC wird die Verbindung zum Atari ST durch einen Druck auf die Return-Taste aufgenommen, und schon arbeiten zwei Computer mit OS-9. Hätte der Atari ST mehrere serielle Schnittstellen, könnten natürlich noch weitere Computer im Timesharing arbeiten. Nachdem wir keine Probleme mit der Verbindung zum Olivetti M24 hatten, versuchten wir, einen mc-68000-Computer unter OS-9 mit dessen Timesharing-Monitor mit dem Atari ST zu verbinden. Auch das funktionierte – nach einigen Basteleien am mc-68000-Computer – zufriedenstellend.
Neben der Übertragung von Textdateien stand auch die Übertragung von Programmen auf unserem Testprogramm. Nach kleineren Anpassungen, die wir hier nicht im Detail beschreiben wollen, liefen einige OS-9-Programme des mc-68000-Computers auf dem Atari ST. Public Domain Software wie z. B. das Kommunikationsprogramm Kermit werden wir auf einer Sammeldiskette anbieten, denn Software für den Atari ST unter OS-9 ist rar und teuer.

Software ist noch Mangelware

Im Lieferumfang des Personal OS-9 ist außer Basic und einem Zeileneditor keine weitere Software enthalten. Ein Assembler mit Linker und Debugger kostet etwa 440 DM, ein C-Compiler schlägt mit 1440 DM zu Buche und ein Texteditor soll für 440 DM zu haben sein. Allein schon durch diese Preise wird OS-9 auf dem Atari ST nicht unmittelbar zu einem Renner werden. Noch etwas teuerer wird es, wenn man sich für fast 3000 DM einen Fortran-77-Compiler zulegt.

OS-9-Basic

In Ermangelung eines besseren muß man sich unter OS-9 mit dem vom 6809-System Dragon 64 her als Basic-09 bekannten Dialekt begnügen, der schon fast zehn Jahre alt ist. Die Basic-Utility wird wie jedes andere OS-9-Programm in den Speicher geladen und ausgeführt. In einer Multiuser-Umgebung können mehrere Anwender gleichzeitig mit dem gleichen Basic arbeiten. Jeder Anwender editiert sein eigenes Programm dabei in einem eigenen Speicherbereich.

```
$ mdir -?
Syntax: mdir [<opts>] [<mod names>]
Function: display module directory
Options:
     -a          print language instead of type
     -e          print extended directory listing
     -t=<type>   list modules only of type <type>
     -u          print unformatted listing

$ mdir -t=prog

    Module Directory at 16:07:28

sysgo       shell       tsmon       Basic       format
mdir        dir         procs       xmode       edt
free        mfree

 Addr     Size      Owner     Perm Type Revs  Ed #  Lnk  Module name
------ --------- ---------- ---- ---- ----  -----  ---  -----------
042516       386       0.0  0555 Prog 0000      4    1  sysgo
0eed20     17540       1.0  0555 Prog 8001     42    2  shell
0ee090      2438       1.0  0555 Prog 8001     14    1  tsmon
0d8120     53702       1.0  0555 Prog 8001     19    1  Basic
0d57d0     10572       0.0  0555 Prog 8001      6    1  format
0d4510      4786       1.0  0555 Prog 8001     16    3  mdir
0d2aa0      6758       0.0  0555 Prog 8001     21    1  dir
0d16c0      5074       1.0  0555 Prog 8001     16    1  procs
0cf980      7474       1.0  0555 Prog 8001     12    1  xmode
0cdf40      6718       1.0  0555 Prog 8001     12    1  edt
0cce80      4274       1.0  0555 Prog 8001     12    1  free
0cc440      2610       1.0  0555 Prog 8001     12    1  mfree
```

Bild 3. Das Kommando „mdir" wird erklärt und zweimal angewandt. Unten wurde mit „mdir -et=prog" die Liste aller installierter Programmodule angefordert.

Das Schreiben von umfangreichen Programmen ist mit dem zeilenorientierten Editor gewöhnungsbedürftig und nicht einfach. Die Funktionen, die der Editor bietet, sind schnell erkärt: Listen von Programmzeilen, direkter Aufruf einer Programmzeile, durch „+" und „-" ändern des Zeilenzählers und mit „c" – dem leistungsfähigsten Editorbefehl – werden definierte Bereiche in der aktuellen Zeile ausgetauscht. Das funktioniert z. B. so, daß man in einer Zeile den Ausdruck „Prnt" durch „c.rn.rin." zu „Print" korrigiert. Unter aktueller Zeile versteht man hier die Zeile, die der Zeileneditor gerade anzeigt. Mit den Cursortasten kann man nicht arbeiten, dafür mit „s" nach Zeichenfolgen in einem Programm suchen lassen.
Syntaktische Fehler erkennt das interaktive Basic gleich bei der Eingabe.

Von Basic-Programmen aus kann man mit dem Shell-Kommando den Befehlsinterpreter des OS-9 aufrufen und z. B. neue Prozesse starten oder Dateien kopieren. Bevor mit dem Microware-Basic Dateien auf Festplatten oder Floppylaufwerken behandelt werden können, müssen sie explizit erzeugt werden. Der Open-Befehl, der das in den meisten Basic-Dialekten automatisch erledigt, reicht hier nicht aus.
Einige Geschwindigkeitstest haben wir durchgeführt, wobei 10000 leere FOR-...NEXT-Schleife etwa 3 Sekunden benötigten. Im Vergleich dazu ist das Omikron-Basic nach 0,24 Sekunden, das GFA-Basic nach 0,4 Sekunden mit dieser Aufgabe fertig. Bei einem weiteren Test lag das OS-9-Basic genau zwischen GFA- und Omikron-Basic. Eine FOR-...NEXT-Schleife von 1...10000 wurde durchlaufen und der Wert des Schleifenzählers bei jedem Durchlauf auf die Festplatte geschrieben. Mit dem Omikron-Basic wurde die Datei in etwa 15 Sekunden angelegt, das OS-9-Basic brauchte etwa 32 Sekunden gegenüber den 44 Sekunden, die das GFA-Basic benötigte. Mit unterschiedlichen Puffer-Speichern ergeben sich etwas andere Zeiten – langsam bleibt es allemal.
Der Festplattentreiber arbeitet zufriedenstellend schnell – im Gegensatz zum Floppylaufwerktreiber. Um die gleiche Datei auf Diskette zu erzeugen benötigte das OS-9-Basic 140 Sekunden; 20 Sekunden mehr als GFA-Basic und 102 Sekunden mehr als Omikron-Basic. Die Diskrepanz zwischen Floppylaufwerk und Festplatte haben wir auch bei der Ausführung von einigen OS-9-Kommandos festgestellt. So ist z. B. ein komplettes Inhaltsverzeichnis der Festplatte mit

```
 Addr     Size      Owner     Perm Type Revs  Ed #  Lnk  Module name
------ --------- ---------- ---- ---- ----- ------ ----- --------------
03a000     15396      0.0    0555 Sys  8000     32     0 kernel
03dc24       212      0.0    0555 Sys  8000      9     1 init
03dcf8       458      0.0    ffff Sys  8000      0     1 stclk
03dec2      1598      0.0    0555 Fman 8000     20     8 scf
03e500      1234      0.0    ffff Driv 8000      1     2 stcio
03e9d2      1416      0.0    0555 Driv 8000      1     3 stsio
03ef5a       120      0.0    ffff Desc 8000      4     2 term
03efd2       122      0.0    ffff Desc 8000      4     3 p
03f04c       118      0.0    0555 Desc 8000      4     3 t1
03f0c2       662      0.0    0555 Driv 8000      0     3 stpptr
03f358       102      0.0    0555 Driv 8000      1     0 null
03f3be       118      0.0    0555 Desc 8000      2     0 nil
03f434      1956      0.0    0555 Fman 8000     24     0 pipeman
03fbd8       102      0.0    0555 Desc 8000      2     0 pipe
03fc3e      5922      0.0    0555 Fman 8000     40     4 rbf
041360      2352      0.0    ffff Driv 8000      1     2 stfd
041c90      1718      0.0    ffff Driv 8000      2     2 sthd
042346       116      0.0    0555 Desc 8000      4     1 d0
0423ba       116      0.0    0555 Desc 8000      4     1 d1
04242e       116      0.0    ffff Desc 8000      4     2 h0
0424a2       116      0.0    ffff Desc 8000      4     0 dd
042516       386      0.0    0555 Prog 0000      4     1 sysgo
0eed20     17540      1.0    0555 Prog 8001     42     2 shell
0ee090      2438      1.0    0555 Prog 8001     14     1 tsmon
0eafc0     10890      1.0    0555 Trap 8002      6     3 cio
0d8120     53702      1.0    0555 Prog 8001     19     1 Basic
0d57d0     10572      0.0    0555 Prog 8001      6     1 format
0d4510      4786      1.0    0555 Prog 8001     16     3 mdir
0d2aa0      6758      0.0    0555 Prog 8001     21     1 dir
0d16c0      5074      1.0    0555 Prog 8001     16     1 procs
0cf980      7474      1.0    0555 Prog 8001     12     1 xmode
0cdf40      6718      1.0    0555 Prog 8001     12     1 edt
0cce80      4274      1.0    0555 Prog 8001     12     1 free
0cc440      2610      1.0    0555 Prog 8001     12     1 mfree
```

Bild 4. Komplettes Verzeichnis aller installierter Module

```
$ procs
 Id PId Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
  2   0   0.0     128   0.26k   0 w        0.00   ??? sysgo <>>>term
  3   2   0.0     128   3.68k   0 w        1.84   ??? shell <>>>term
  4   0   0.0     128  10.68k   0 s        0.00   ??? tsmon <>>>t1
  5   3   0.0     128  18.95k   0 a        0.14  0:00 mdir <>term >p
  6   3   0.0     128  15.12k   0 *        0.56  0:00 procs <>>>term
```

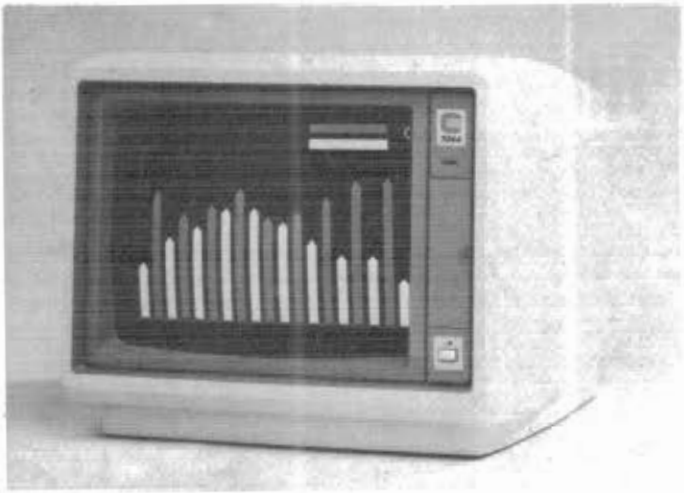
Bild 5. Eine typische Prozeßauflistung

„dir /h0 -r >verzeichnis" in etwa 20 Sekunden (8 Sekunden mit einem Device-Deskriptor, der 256 KByte Puffer-Speicher verwaltet) erstellt. Für die gleiche Aufgabe mit der Systemdiskette in Laufwerk /d0 brauchte OS-9 etwa 86 Sekunden (31 Sekunden mit 256 KByte Puffer-Speicher). Billigt man sowohl Festplatte als auch Floppylaufwerk 256 KByte Puffer zu, wird es auch auf einem 1-MByte Atari ST eng. Etwas Speicher (64 KByte oder 256 KByte) sollte man sich für eine RAM-Disk freihalten, die durch Einbinden des entsprechenden Device-Deskriptors in das Bootstrap-File initialisiert wird.
Die Bedienung des OS-9-Basic und seine Leistungsfähigkeit verdienen kein besonderes Lob, weswegen wir hier nicht weiter darauf eingehen wollen. Lediglich die Möglichkeit, mehrere Prozesse parallel laufen zu lassen, macht dieses Basic interessant.

OS-9 – und was weiter?

Resümee: Zur Programmierung in Basic lädt das System nicht ein und Programmiersysteme für andere Sprachen kosten viel Geld. Auf alle Besonderheiten wie Grafik, Accessories, Farbe und GEM, die den Atari ST so reizvoll machen, muß man unter OS-9 verzichten. An einer Implementation von GKS soll wohl gearbeitet werden. Ein Zeileneditor, der vor Jahren vielleicht „state of the art" war, ist heute kaum mehr akzeptabel.
Mit einem bildschirmorientierten Editor und weiteren Utilities könnte man das bestehende OS-9 erheblich aufwerten. Nach unserem kleinen Exkurs in Sachen OS-9 bleibt festzustellen, daß Atari ein Multiuser-/Multitasking-Betriebssystem für den Atari ST verfügbar hat, für das sich hauptsächlich Anwender interessieren werden, die das Betriebssystem von anderen Rechnern her kennen und jetzt einen preiswerten Zweitrechner wollen.
Alle Programmierer sind aufgefordert zu zeigen, was sie können, damit sich das leistungsfähige Betriebssystem auf dem Atari ST durchsetzt. Und das wird nur dann der Fall sein, wenn es auch Anwendungssoftware gibt, mit der die Vorteile von OS-9 genutzt werden können. Bis dahin wird es da bleiben wo es ist – im Elfenbeinturm.

An Schulen und Bildungseinrichtungen jedenfalls wird man sich über das preiswerte Unix-ähnliche System freuen.



EGA-Farbe im stattlichen Großformat auf den neuen Conrac-Monitoren

Großflächen-Farbe

Fast schon im TV-Maxi-Format präsentiert sich der 19-Zoll-EGA-Farbmonitor Conrac 7164, ausgerüstet mit einer hochauflösenden Inline-Bildröhre (0,31 mm Lochabstand). Als Videobandbreite spezifiziert das Datenblatt 40 MHz, ein Wert, den man bei gehobenen Ansprüchen durchaus braucht, denn es müssen immerhin 640 × 350 bzw. 640 × 200 Bildpunkte (je nach Betriebsmodus der EGA-Karte) in ansehnlicher Brillanz auf der Schirmfläche abgebildet werden.

Ein kleiner Bruder dieses Monitors nennt sich 7064, er arbeitet mit 23 MHz Bandbreite, bietet einen 14-Zoll-Bildschirm und besitzt eingangsseitig – wie das größere Modell – die RGB-TTL-Schnittstelle.

Conrac Elektron GmbH
Postfach 60
6992 Weikersheim
☎ 0 79 34/10 10

Software-Kompendium

Kostenlos an alle Interessenten abgegeben wird der neue Softwarekatalog von Feltron/Zeissler, der auf über 80 Seiten die Programmpakete fast aller wichtigen in- und ausländischen Softwarehäuser enthält. Zugeschnitten auf die Betriebssysteme CP/M und PC-DOS enthält die Angebotspalette die Produktbereiche Betriebssysteme, Assembler, Programmiersprachen, Datenbank-, Tabellenkalkulations- und Textprogramme sowie diverse System-, Apple- und Spezial-PC-Software.

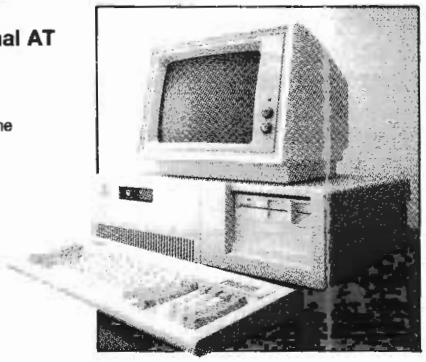
Alle in diesem Katalog zusammengestellten Programme können zur Zeit auf über 132 verschiedene andere Diskettenformate umkopiert werden, wodurch sich viele Kompatibilitätsprobleme lösen lassen.

Feltron Elektronik
Postfach 12 63
5210 Troisdorf-Spich
☎ 0 22 41/4 10 01

Choose it for the Best!!!

Com Trade Turbo XT
- 8088-2 PC/XT-kompatibel
- 4,77/10,0 MHz, bis zu 640 KByte RAM auf Hauptplatine erweiterbar
- 6-Pak Multi I/O-Karte
- Monochrome Grafikkarte
- Tastatur; 150 W
- Gehäuse mit Schlüsselschalter und LED
- 8087 Coprozessor als Option

Com Trade Professional AT
- 80286 PC/AT-kompatibel
- 6/10 MHz, 6/12 MHz, 0 oder 1 Wortezyklus
- 1 MByte RAM auf Hauptplatine
- Monochrome Grafikkarte (Jumper switch)
- FDD/HDD Controller
- Seriell-/Parallelkarte
- XAT-Tastatur; 200 W
- AT-Hauptplatine in XT-Format auch verfügbar



WIR SIND TAIWANS FÜHRENDER GEHÄUSEHERSTELLER

Wir machen alles: Vom Stanzen und Pressen bis zum Lackieren kontrolliert Ling Yih jeden einzelnen Schritt der Herstellung. Wir zeigen Ihnen gern unsere Fabrik – Ling Yih hat nichts zu verbergen, aber viel zu zeigen. Wir bieten nicht nur höchste Qualität, sondern auch einen annehmbaren Preis. Schreiben Sie uns noch heute. Wir liefern prompt.

ComTrade Technology Co., Ltd. LING YIH CO., LTD.
9F-3, 57 FU-HSING N. RD., TAIPEI, TAIWAN, R.O.C. TEL: (02)771-7166 FAX: (02)711-3506 TLX: 20232 LINGYIH

Echte 32 Bit unter Unix V: CPS-Rechnerserie von Cadtronic

32-Bit-Power im Kompaktformat

Herz der Cadtronic-32-Bit-Rechner sind die ICM-Boards von National Semiconductor, die jetzt in einer noch leistungsfähigeren Version erscheinen: Die bisherige 3216-Karte wird jetzt abgelöst durch das Prozessorboard ICM 332-1, das mit dem kompletten 32000-Chipsatz ausstaffiert ist und unter der CPU 32332 arbeitet. Ein durchgängiges 32-Bit-Konzept ersetzt damit die bisherige 32/16-Bit-Architektur; als I/O-Prozessor findet sich übrigens der 32016.
Die mit der genannten Hardware aufzubauenden Rechner der CPS-Serie sind zugeschnitten auf das Standard-Betriebssystem Unix V und erlauben dem Anwender den Zugriff auf eine breite Softwarepalette aus dem kommerziellen bzw. technisch/wissenschaftlichen Bereich.

Cadtronic Computer Systeme GmbH
Britanniahütte 10
5060 Bergisch Gladbach 2
☎ *0 22 02/5 20 42*

Basic — das Wahre

True Basic, so nennt sich ein Interpreter/Compiler aus den USA, lauffähig auf IBM-PCs und Kompatiblen, auf Apple Macintosh, Commodore Amiga und Atari ST (Programme kann man, da im Quellcode, zwischen diesen Systemen beliebig tauschen). Vorteile von True Basic sind: Programmieren in strukturierter Form ohne Zeilennummern, eine Reihe neuer, leistungsfähiger mathematischer und trigonometrischer Funktionen, verschiedene 2 D-Grafikfunktionen oder ein 3 D-Paket (als Ergänzung). Interessant für den Elektroniker: Mit einer Zusatzkarte wird der Zugriff auf den IEC-Bus (IEEE 488) möglich; man kann somit via Basic einen automatischen Meßplatz steuern.
Jürgen Pfotenhauer
Neulandstr. 16
7590 Achern
☎ *0 78 41/50 56*

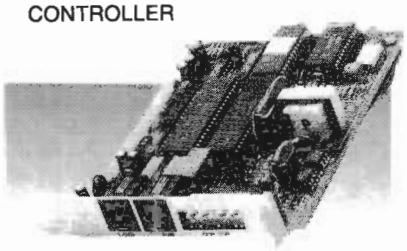
BEST MODEM Just You Need!

Sole Agent Wanted

SPECIAL DESIGN FOR CCITT STANDARD

BEST 1200 PLUS:
* CCITT V.21/V.22/V.23 COMPATIBLE
* INTERNAL BUFFER FOR V.23 SPLIT BAUD RATE (8K MAXIMUM)
* RS-232C INTERFACE BAUD RATE: 300/1200, 1200/300
* MODEM BAUD RATE:
  V.21:0-300 FULL DUPLEX
  V.22:1200 FULL DUPLEX
  V.23: 75/1200, 1200/75 ASYMMETRICAL FULL DUPLEX
  1200/1200 HALF DUPLEX

BEST BABY AT:
* 5 CUSTOMER CHIP
* 80286 CPU WITH OPTION (80287 MATH CO-PROCESSOR)
* 6/8MHz, 6/10MHz HARDWARE/ SOFTWARE SELECTABLE
* 0K-1MB MOTHER BOARD
* 4 LAYER PC MOTHER BOARD
* COMPACT SIZE
* 1.2MB FLOPPY DISK CONTROLLER
* 2.5MB MULTIFUNCTION CARD (SERIAL/PARALLEL/GAME)
* 20MB OR 40MB HARD DISK & CONTROLLER CARD
* 200W POWER SUPPLY
* ON BOARD NI-CD BATTERY FOR REAL-TIME
* EGA CARD OR MONOCHROME HERCULES

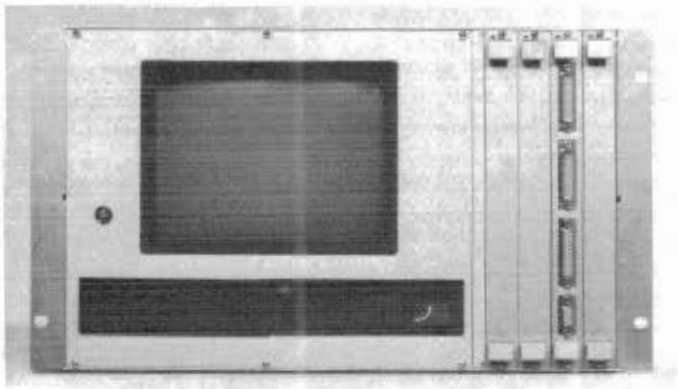


BEST 1200B:
* INTERNAL MODEM CARD FOR IBM PC/XT/AT OR COMPATIBLE
* SAME GREAT FEATURES AS THE BEST 1200 PLUS

BEST 1200 PLUS:
* HAYES COMPATIBLE
* CCITT V.21/V.22 & BELL 103/212A
* AUTO-DIAL, AUTO-REDIAL, AUTO-ANSWER
* DATA/VOICE SWITCHABLE (OPTION)
* AUTO-SPEED SELECTION
* CALL PROGRESS DETECTION
* REMOTE DIGIT LOOPBACK TEST
* RS-232C PORT

BMODEM
BEST COMMUNICATION INC.
No. 191, 2F-3, Fu Shing N. Road, Taipei, Taiwan, R.O.C.
Telex: 29690 BESTCOMM Tel: 886-2-7174088 Fax: 886-2-7163703

HAYES is registered trademark of Hayes Microcomputer Products Inc.
IBM PC/XT/AT are registered trademarks of International Business Machines Corp.



Die adps-Computer für den industriellen 19-Zoll-Einbau können „nach Maß" zusammengestellt werden

Industrie-PC als Baukasten

Eine breite Palette an PC/XT/AT-kompatiblen Boards für den Einbau in 19-Zoll-Chassis führt adps in ihrem Programm; mit ihnen kann der Anwender sich seinen Computer, der dann später auch härteren Umgebungsbedingungen gerecht wird, maßgeschneidert selbst zusammenstellen und sich damit eine Alternative zum Computer „von der Stange" bilden. Ergänzung findet das Angebotsspektrum durch Silicon-Disk-Controller, Netzwerk-Interfaces sowie durch 32-Bit-Boards für höhere Geschwindigkeiten bei komplexen Applikationen. Die dazugehörende Betriebssoftware wird auch Echtzeit- und Multitasking-Anforderungen gerecht.

Advanced Data Processing Systems
Gutenbergstr. 14
7024 Filderstadt 4
☎ 07 11/77 20 49

Den Apple aufmöbeln

Wer viel mit der NF-Technik zu tun hat und gerne seinen guten alten Apple II hierzu mitverwenden möchte, der sollte sich für ein wirklich als universell zu bezeichnendes Signalsystem interessieren, das als Zusatzplatine recht einfach zu installieren ist und im Frequenzbereich von 10 Hz...38,5 kHz die Signalspeicherung und -manipulation, die Kennlinienaufnahme, Frequenz- und Impulsmessungen sowie auch die Logikanalyse ermöglicht. Das allein sind schon sehr beachtliche Funktionsmerkmale, aber die mit der Erweiterungsplatine gekoppelte Software erlaubt darüber hinaus noch die Fast-Fourier-Signalanalyse und als besonderen Clou die Signalgenerierung, wobei alle mathematisch definierbaren Funktionen als Analog-Ausgangswellenformen erzeugbar sind. Alle eben genannten Betriebsarten werden vom Bildschirm her via Menü aufgerufen – Einarbeitungszeit entfällt.

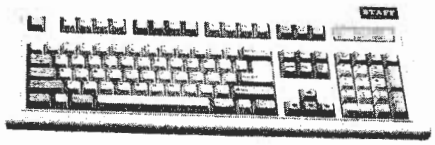
Ingenieurbüro U. Karrenberg
Mintarder Weg 90
4000 Düsseldorf 30
☎ 02 11/42 07 64

"STAFF" – A brand name built on high quality & prompt delivery

STAFF-2P AT

12MHz

STAFF-2P AT
COMPACT DESIGN, HIGH PERFORMANCE!
This fantastic "BABY AT" is a well-developed system with enhanced compact hardware and fully PC/AT software compatible. The STAFF-2P AT is surely an ideal choice for any imaginable computing need.



STAFF-K101 **STAFF-K101**
FCC ID: FKD46AK101
* 101 keys American/102 keys European
* XT/AT compatible
* 12 function keys

STAFF-K9AX

STAFF-K9AX
FCC ID: FKD46AK9AX
* 105 keys
* XT/AT compatible
* Special functions keys: Key-in lock; Fast repeat; Previous word; Next word; Line feed; Pause; Clear screen; Reset; Break

MONTEREY International Corp.
P.O. Box 55 – 850, Taipei, Taiwan, R.O.C. Office: 1-6 F No. 40, Deh Hwei Street, Taipei, Taiwan. Phone: 886-2-5917138. Cable: MONTEREY TAIPEI. Telex: 25171 MONTEREY. Fax: 886-2-5931075.

Schon als „Schnittstellen-Prüfstand" zu bezeichnen ist das kompakte Testgerät von BEC

Daten-Kontrolleur

Die auf maximal sechs asynchronen, seriellen Schnittstellen (davon zwei TTY-Schleifen und vier V.24-Leitungen) ausgetauschten Daten kann der kompakt aufgebaute Schnittstellentester der Fa. BEC dokumentierend unter die Lupe nehmen.
Auf der Flüssigkristall-Anzeige darstellbar ist natürlich nur jeweils einer dieser Datenströme, dafür jedoch in einem frei wählbaren Format. Als Pufferspeicher fungieren 24 KByte, deren Inhalt nach einer Übertragungssequenz in aller Ruhe auf dem Display betrachtet und nach Fehlern abgesucht werden kann. Recht praktisch ist der eingebaute Editor zum Erstellen eines Testprogrammes, das Datenfehler analysieren, Zeichenfolgen erkennen und die Aufzeichnung beim Erreichen eines vorwählbaren Zählerstandes stoppen kann. Gleichermaßen lassen sich damit in Abhängigkeit von den Eingangsdaten beliebige Zeichenketten an das Testobjekt abschicken. Für den, der alles schwarz auf weiß haben möchte, steht eine PC- und Drucker-Schnittstelle zur Verfügung.

Becker-Electronic-Computer GmbH
Hölzlestr. 32
7460 Balingen
☎ 0 74 33/83 31

Die richtige Wacker-Adresse

In unserer Übersicht über PC-Erweiterungskarten in mc 4/1987 veröffentlichten wir auf Seite 59 kurze Produktbeschreibungen über den Fourierprozessor PC-FFT und den Datenakquisitions-Prozessor DSP-16 der Wacker GmbH. Durch ein Versehen wurde jedoch die Adresse nicht richtig wiedergegeben. Sie muß korrekt lauten:
Wacker GmbH
Niedenau 36
6000 Frankfurt
☎ 0 69/72 50 45

Unsere Preise machen Ihre Computer konkurrenzfähiger
Mit uns sind Sie führend

COME WITH US, YOU WILL ALWAYS BE THE LEADER

LEGAL

6/8/10/12 MHz SWITCHABLE BOARD
PC-3200 AT COMPATIBLE COMPUTER

- Mini AT Board oder Standard AT Board
- 80286 CPU, 8 slots, 512K/640K/1024K/(option) 6/8/10/12 MHz Switchable.
- Color Graphics Printer Card, Monochrome Graphics Printer Card Enhance Graphics Adapter (option).
- One Serial and one parallel and one Game
- One 1.2MB Floppy Disk Drive.
- Hard Disk/Floppy Disk Controller.
- 20MB Hard Disk.
- 101 Keys Keyboard.
- 200W Power Supply, 110V/220V Switchable

16 BIT PERSONAL COMPUTER
PC/XT COMPATIBLE

- Keyboard available in english, German, French, Spanish, Danish, and other language.
- Color or mono/graphics card selectable.
- Supports 10MB or 20MB hard disk.

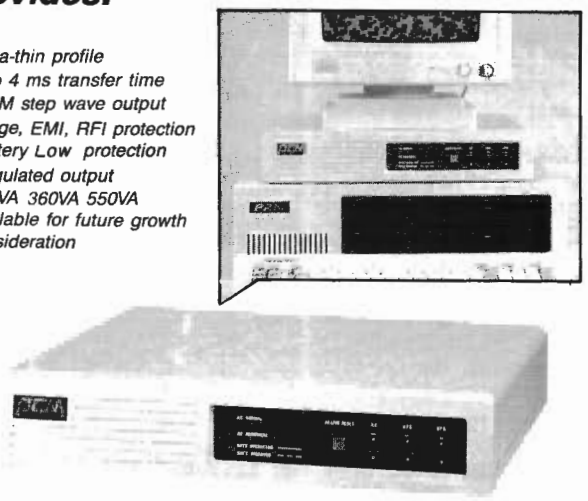
ADD-ON CARDS FOR PC/XT/AT: MINI AT Board

- 640K Main Board
- Turbo Board
- Multi I/O Card
- Mono/Graphic/Printer Card
- EGA Card
- Color/Mono/Graphic/Printer Card
- 80286 Speed Card
- D-Link Card
- Multi User Card (for AT)
- IEEE 488 Card
- 3MB Multifunction Card
- Serial/Parallel Card
- Color/Graphic/Printer Card
- And many more!

P & C SHITEN ENTERPRISE CO., LTD.
RM 7, 8FL., No 100, ROOSEVELT RD., SEC 3, TAIPEI, TAIWAN, R.O.C.
CABLE: PCSHITEN TLX: 19206 PCSHITEN TEL: (02) 395-1400
P.O.BOX: 30-291, TAIPEI, TAIWAN, R.O.C. FAX: 886-2-3512073

NOT UPS ONLY, IT OFFERS MORE...
ULTRA — THIN UPS = Power Center + Uninterruptible Power Supply, backs up PC perfectly, provides:

- *Ultra-thin profile*
- *1 to 4 ms transfer time*
- *PWM step wave output*
- *Surge, EMI, RFI protection*
- *Battery Low protection*
- *Regulated output*
- *250VA 360VA 550VA available for future growth consideration*

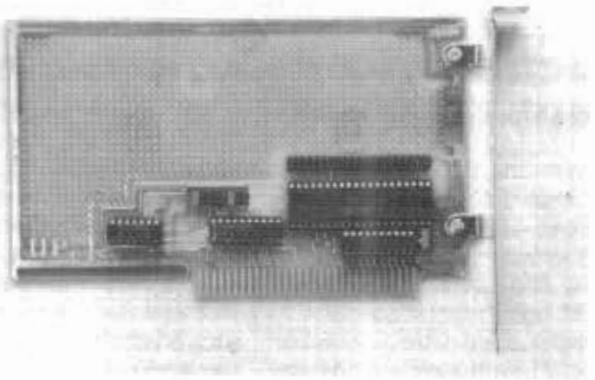


POWERCOM CO., LTD.
No. 4, Alley 18, Lane 252 Yuan Tung Rd., Chung Ho City
Taipei, Taiwan, R.O.C.
Telex: 32585 POWERCOM Tel: (02)9447453 Fax: 886-2-9447455

Eigenbau für den PC

Eine Prototypenkarte von Kolter Electronic eignet sich bestens für kleinere Eigen-Entwicklungsarbeiten am PC: Sie besitzt einen 8255 PIO-Baustein und kann somit direkt 24 I/O-Leitungen ansprechen. Steckbrücken sorgen dabei für die wahlfreie Adressierung. Genügend Platz für kleinere Entwicklungsarbeiten bietet das Lochrasterfeld der Karte, das am Rand gleich die Versorgungsspannungs-Leitungen in die Nähe der Bauelemente bringt. Kostenpunkt: 198 DM einschließlich informativem Basic-Demo- und Help-Programm.

H. Kolter Electronic
Postfach 11 27
5042 Erftstadt
☎ 0 22 35/7 67 07

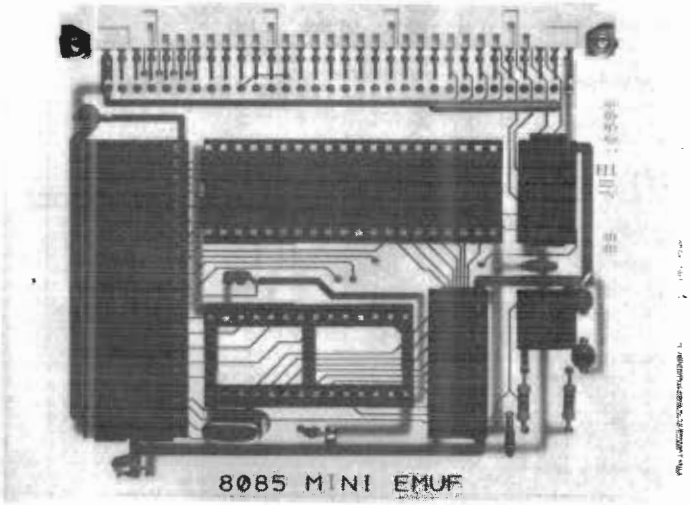


Ideal für kleinere Hardware-Erweiterungen am PC ist die Prototypenkarte der Fa. Kolter

Der Mini-EMUF

Auf einer halben Europakarte hat das Ingenieurbüro W. Kanis eine weitere Version der Einplatinencomputer-Serie EMUF untergebracht, und zwar den „8085 Mini" mit der gleichnamigen CPU (2 MHz Takt), dem Port-Baustein 8155 (16 I/O-Kanäle), maximal 16 KByte EPROM, mit 128 Byte RAM und einem 14-Bit-Zähler/Zeitgeber. Aufhorchen läßt der Preis: Er liegt bei 100 DM.

Ingenieurbüro W. Kanis GmbH
Lindenberg 113
8134 Pöcking
☎ 0 81 57/35 76

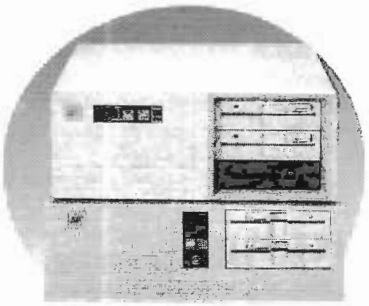


Preisgünstiger „Halbplatinen"-EMUF: 8085 Mini

You'll Never Regret to Contact Us

ST386
- CPU: 80386 w/80387 socket
- 512K RAM interleaved 32-bit RAM on board
- 1.2MB floppy disk drive
- FDD/HDD controller
- 101 keys keyboard (FCC approved)
- Enhanced graphic adapter

OEM Welcome!



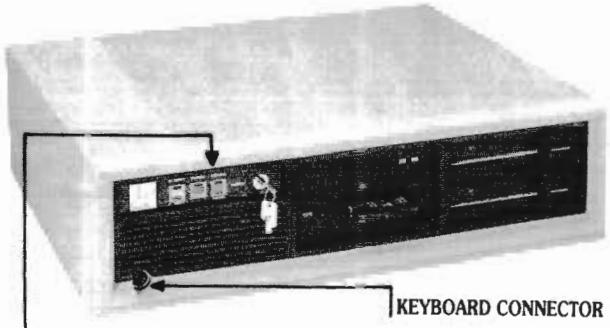
ST300A BABY AT (6/10/12MHz switchable)
- CPU: 80286 w/80287 socket
- 1MB RAM expandable
- Dimension: 42.5(D) x 44.0(W) x 14.5(H) cm

ST300B BABY AT (6/8MHz switchable)
- CPU: 80286 w/80287 socket
- 1MB RAM expandable
- Dimension: 43.0(D) x 41.0(W) x 15.5(H) cm

CONTACT US NOW FOR 6/12/16 MHZ

SUPER TRACK TECHNOLOGY INC.
11F., No. 491, Min Sheng E. Rd. Taipei, Taiwan, R.O.C.
Tel: (02)5058182 Telex: 11370 CHIPSINC
Fax: 886-2-5012363

386 SYSTEMS



- KEYLOCK, RESET, POWER, FDISK
- '0' & '1' WAITSTATE SELECT SW.
- 6,8 & 10 MHZ TURBO SELECT SW.
- FULL LIGHT INDICATION.

386 PROFESSIONAL CASE:
CAN MOUNT WITH 386, 286 (PC/AT, BABY AT) SYSTEM BOARD, FRONT PANNEL (CONTROL SWITCH) CAN CHANGE DEPEND TO YOUR SYSTEM, SAFE MOUNTED FOR FDD, HDD

WE OFFER:
- 386 PROFESSIONAL CASE.
- 386 SYSTEM, PC/XT/AT COMPATIBLE SYSTEM
- ALL INTERFACE CARD, ETC...

Manufacturer & Exporter
JCL JOLLITY
ELECTRONICS INDUSTRIAL INC.
2F., 180, Chung Ching N. Rd., Sec. 4, Taipei 11161, Taiwan, R.O.C.
Telex: 14831 JOLLITY Tel: (02)8116249 . 8119869
Fax: 886-2-8114467 Atn: JCLJOLLITY Cabl: JOLLITY Taipei

Sieht gut aus, ist gut, und kostet wenig:

COMMODORE PLUS/4

Postvertriebsstück
B 7745 E

6710 FRANKENTHAL

64 K Speicher, 121 Farben, 4 Tongeneratoren, hervorragende Grafikmöglichkeiten. Anschließbar an normalen Fernseher- oder Monitor (s.u.). Programmierbar im erweiterten BASIC 3.5 mit vielen neuen Befehlen. **Und es kommt noch besser:** 4 fertige Software-Programme sind bereits fest eingebaut: Textverarbeitung mit vielen Möglichkeiten, die viel teurere Textsysteme auch bieten; Tabellenkalkulation zur Berechnung umfangreicher Zahlenkolonnen; Dateiverwaltung für Adressen, Kunden, Lagerartikel oder andere Daten; Geschäftsgrafik zur optischen Aufbereitung von Daten.

...für den ...welt ein... ...chkeiten ...al für al... ...tern be... ...schäftigen wollen!

MONITOR nur 129.-

Links:

FERGUSON

Video-Datenmonitor von THORN EMI. Unglaublich gut für diesen Preis: 12 Zoll großer Bildschirm, 25 Mhz, Videoeingang mit CHROMA/LUMA, dadurch besonders für COMMODORE-Heimcomputer geeignet. **129.-**

Gleicher Monitor m. bernsteinfarbenem Bildschirm **139.-**

PLUS/4 komplett mit FERGUSON-Monitor (grün) **299.-**

PLUS/4 komplett mit Floppy 1551 und FERGUSON-Monitor (grün) **499.-**

PLUS/4 incl. Datenrecorder u. BASIC-Lernkurs **229.-**

Diskettenstation (Floppy) 1551 zu PLUS/4, 170 K ~~299.-~~

PLUS/4 komplett mit Floppy 1551 **399.-**

TURBO-PLUS Erweiterungsmodul Turbo-Tape, 14 neue Befehle **49.-**

UNIVERSELL: FLOPPY VC 1541 für C 16, PLUS/4, C 64, C 128 nur **399.-**

COMPUTER PLUS/4 nur 198.- (ohne Monitor)

AB 6. JUNI AUCH IN 4300 ESSEN HUYSSENALLEE 3

Das VOBIS-Ballon-Team ist mit von der Partie und verlost Freifahrten!

kompetent + preiswert

VOBIS

Deutschlands umsatzgrößter Microcomputer-Spezialist

WICHTIG! Herstellerbedingte Lieferzeiten! Aufgrund erhöhter Nachfrage ist nicht immer alles sofort lieferbar!

VERSAND-ZENTRALE:
Postfach 1778
Rotter Bruch 32-34
5100 AACHEN
☎ 0241/500081
Tx 832389vobis

FILIALEN:

AACHEN Viktoriastr. 74 · 0241/54 31 00
BERLIN 30 Kurfürstenstr. 101 · 030/2 13 94 80
BIELEFELD Herforder Str. 106 · 0521/6 38 78
BREMEN Violenstraße 37 · 0421/32 04 20
DORTMUND Hamburger Str. 110 · 0231/57 30 72
DÜSSELDORF Wielandstr. 21 · 0211/35 99 64
FRANKFURT Frankenallee 207/209 · 069/73 40 49
HAMBURG Krohnskamp 15 · 040/2 79 46 76
HANNOVER Berliner Allee 47 · 0511/81 65 71
KARLSRUHE Kriegsstr. 27/29 (am BGH) 0721/37 82 68
KONSTANZ Kreuzlinger Str. 18 · 07531/1 55 60
KÖLN Mathiasstr. 24-26 · 0221/24 86 42
MÜNCHEN Aberlestr. 3 · 089/77 2110
NÜRNBERG Vordere Ledergasse 8 · 0911/23 29 95
STUTTGART Marienstr. 11-13 · 0711/60 63 36